電腦模擬在科學與工程上的重要性

學物理有什麼用？學了物理的人能做什麼事？

大學之道，在明明德，在親民，在止於至善（全文）。我的一位高中老師說，唸大學的目的是明理，另一位說也在改變氣質。我的博士論文指導老師說，有一個他以前學生去應徵工作，主管問他你能做什麼，他回答：**解決問題**（"I solve problems."）。

我們學物理除了可以滿足自己的好奇心，還可以建立一個很重要的能力，就是解決（解答）問題的能力。解決問題的能力來自於預測因果關係，知道把以什麼角度力道球拋向空中，它會落在那裏，這種能力的延伸，讓人類可以把人造衛星打到遼遠的行星上。把多少放射性元素放在一起，連鎖反應就可以持續不斷產生核能。

其他的小故事：怎應樣飛機引擎輕一點，以便節省燃料。表面摩擦學泰伯 (Tabor) 的小故事。汽車碰撞模擬。

新典範：電腦模擬

電腦模擬繼觀測與實驗之後，成爲第三種人類確立自然科學知識的典範（paradime）。電腦成爲協助解決科學研究與工程應用問題上的一個不可或缺的工具。

以電腦協肋我們解決問題的方式，可參見我提出的金字塔：

問題
物理量
數學模型
演算法及程式
數值計算及分析
結論、解釋、預測

我的數值方法課包含了上述金字塔裏頭：演算法、程式、數值運算、分析。其中數值運算包含運算環境的建構與運算操作的實作，而分析則包含數據的整理呈現與作圖。

科學與工程運算的例子

電磁場與電磁波模擬（電磁波相容性分析）

電路設計模擬

工程建築結構分析

汽車碰撞模擬

流體力學（如噴墨印表機）與空氣動力（風洞）模擬

氣象預報

分子動力學模擬（材料物性與溫度的關係）

藥物設計（鑰匙與鎖孔配對的原理）

量子力學能帶結構與材料物性計算

高能粒子場論模型計算

石油探勘與地震即時資訊處理

資料視覺化與虛擬實境（斷層掃描）

基因密碼解析

找尋外星人（無線電波天文學）

科學資料庫與搜尋介面 (ICSD, CSD, PDB)

隱形飛機的雷達截面積(RCS)模擬

http://home.xmsnet.nl/hdejong/curious/Lampyridae.htm

一個提到科學運算種類的網站資料：

http://www.csc.fi/english/csc/scientific_computing/applications

大學之道全文及評注：

大學之道：在明明德，在親民，在止於至善。知止而后有定，定而后能靜，靜而后能安，安而后能慮，慮而后能得。物有本末，事有終始，知所先後，則近道矣。

古之欲明明德於天下者，先治其國；欲治其國者，先齊其家；欲齊其家者，先脩其身；欲脩其身者，先正其心；欲正其心者，先誠其意；欲誠其意者，先致其知；致知在格物。物格而后知至，知至而后意誠，意誠而后心正，心正而后身脩，身脩而后家齊，家齊而后國治，國治而后天下平。

自天子以至於庶人，壹是皆以脩身為本。其本亂而末治者否矣；其所厚者薄，而其所薄者厚，未之有也。

子吉評曰：《大學之道》乃是《四書》中的第一篇文章，文中提出「明德格物」和「修身爲本」的精神，非常值得我們注意和警惕。同時文中亦提醒我們，需要認識事物的本末先後，才可以在學業上有成績。可惜的是，現時的學生大多數都缺乏獨 立而又清晰的思考訓練，對於事物的「本末」未能掌握，引致經常犯上「本末倒置」的錯誤。

請你跟我這樣做：快速入門

學習策略

模仿是學習的有效途徑之一，透過一步步跟著做，讓大家以最快的速度先學 "一招半式"，從先求有、再求精，最後再通。

學習目標

一、使用 VMPlayer 下的 Linux 作業系統，能啓動與關閉。

二、最基本的 UNIX 指令、以 vi 觀看並試著理解三個附屬範例程式的內容、並編譯 (compile) 及執行這些程式。

三、以 vi 文字處理器撰寫簡單的 fortran 程式、能計算的 fortran 程式，以及能畫圖的 fortran 程式。

四、知道要到何處找尋更多關於 UNIX 指令、vi 功能、fortran 指令，以及 pgplot 繪圖指令。

快速入門開始

快速入門

啓動 VMWare Player 來播放預先灌好的 Linux 作業系統之虛擬機器。

從 Windows 的開始 

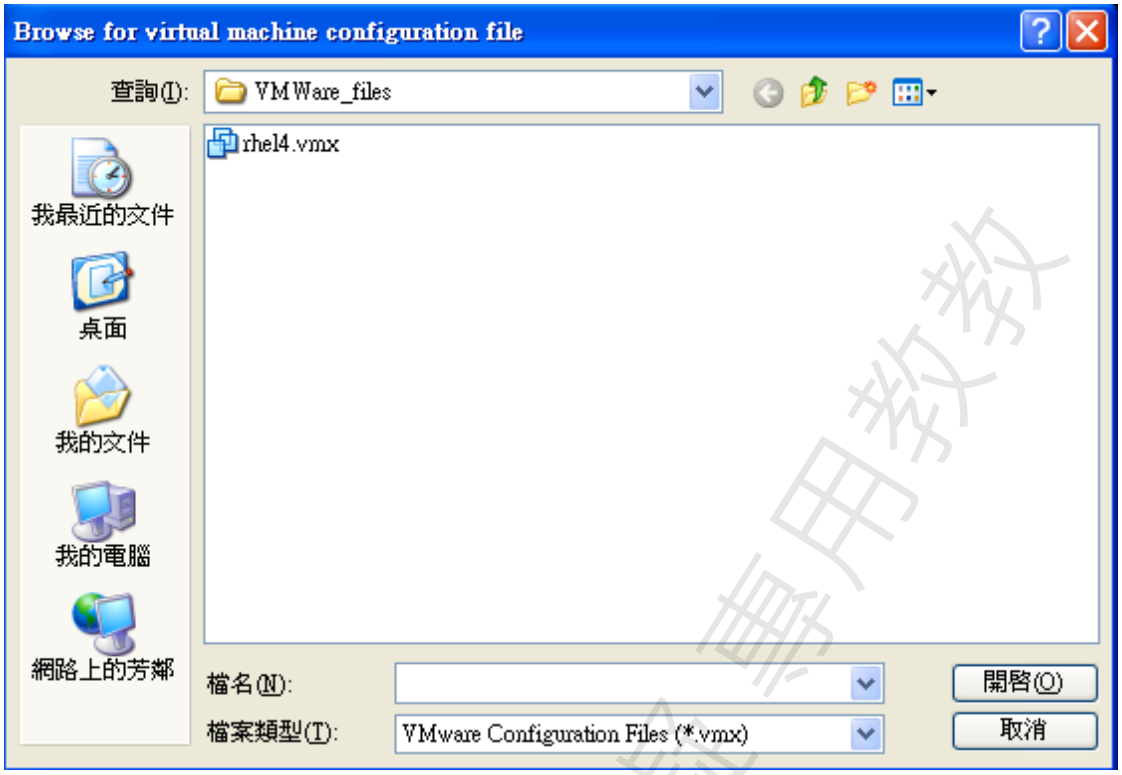
其中的 

，

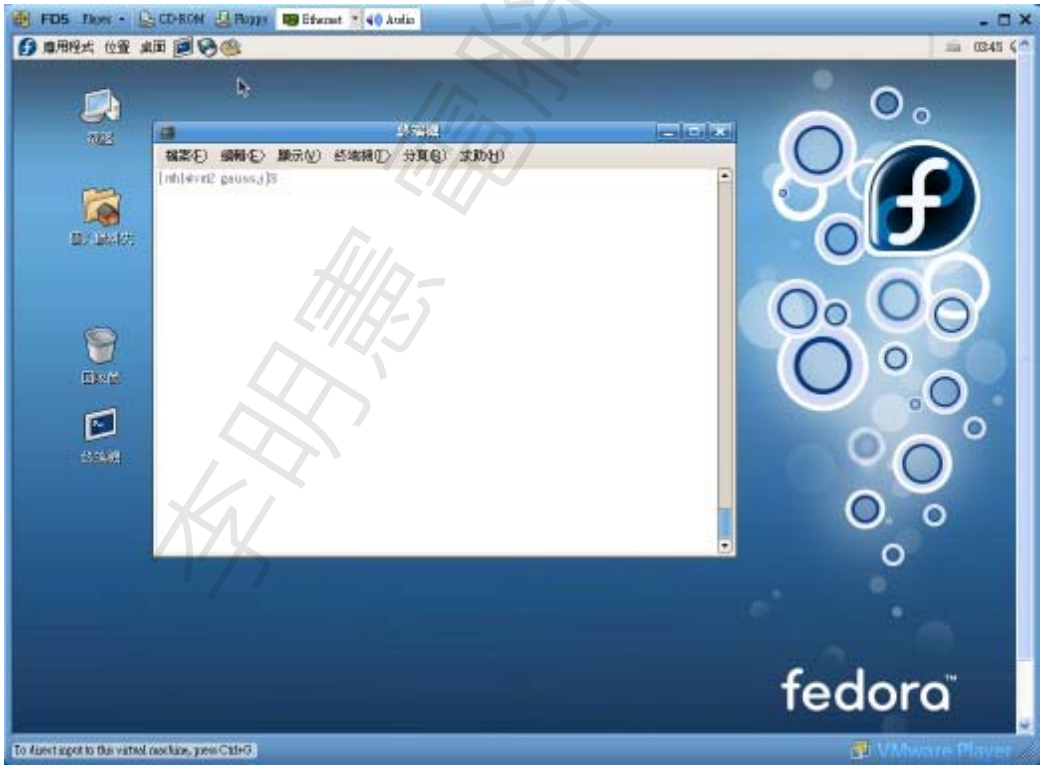
找尋 VM Ware 目錄，點選裏面的 VMware Player

如果是最近有使用過 VMware Player，則也可以直接從 開始 的 選單中起動

開啓 VMware Player 之後，會詢問虛擬機器的影像設定檔在那裏



選好後按 [開啓]，則會喚醒虛擬機器（如果前一次在使用後是做休眠的動作的話），回復到之前使用之狀態的最後畫面。



如果前一次是虛擬機器的關機，則是完全相似於一般電腦開機的畫面

登入系統（若已登入則可跳過）

開機訊息跑完之後，會進入 UNIX 之 X-視窗的圖形登入介面

以 `student` 爲帳號並以 `student` 爲密碼來登入

密碼接受後，會花一些時間作載入的動作，

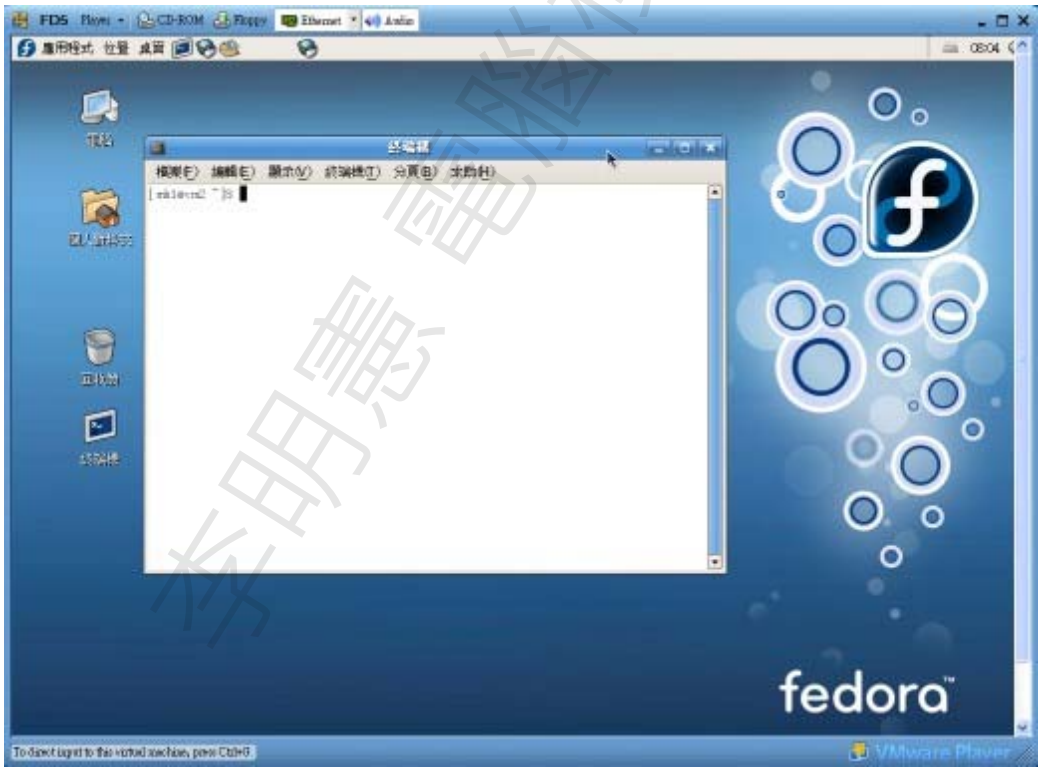
然後進入 X-window 桌面環境

開啓一個終端機視窗

再下一步，我們就要開啓一個終端機視窗，其實它的捷徑已經在桌面以及上排 menu bar 都有了（看起來像一個螢幕的圖示），至少最標準的方法，是從應用程式選單上去選

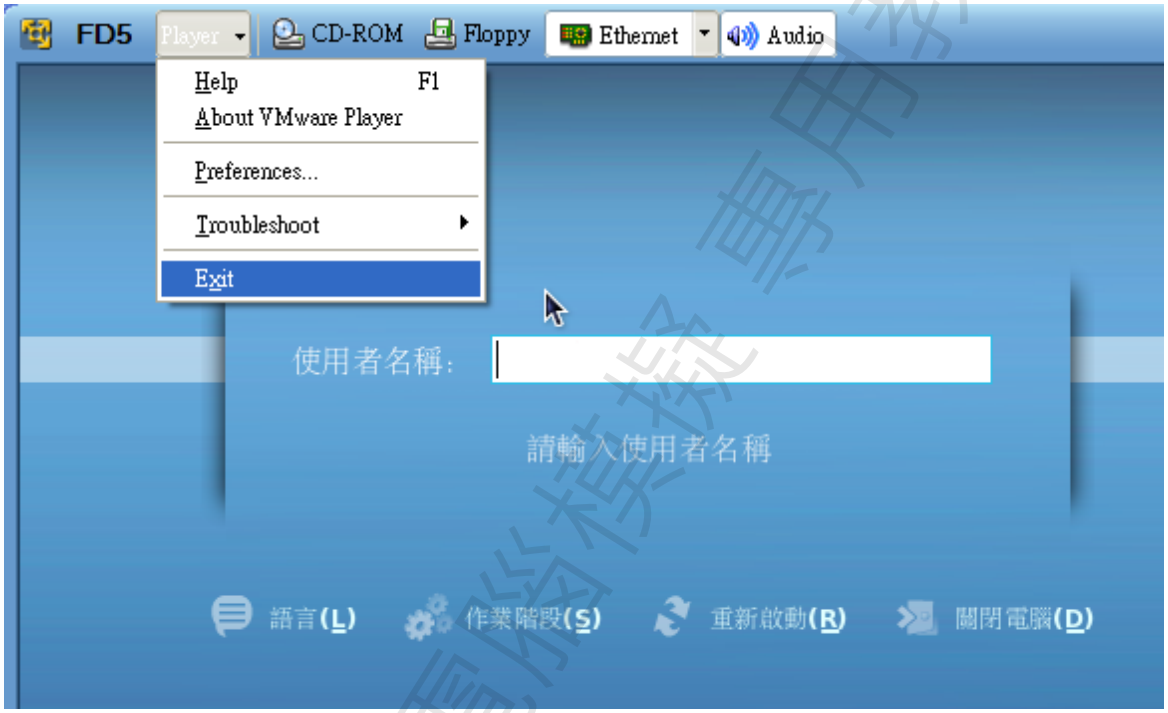
請注意，在 Fedora 11 版本的 Linux 桌面，終端機是收在 "應用程式" 類的 "系統工具" 之內。

如此，就會有一個終端機視窗被開起來



滑鼠游標要從 VM 中跳出來回到 Windows 桌面的環境，則同時按 Ctrl 及 Alt 兩個鍵，事實上在 VM Player 的訊息欄（左下角）也會有提示。

在操作過程中的任何階段停止虛擬機器的方法，



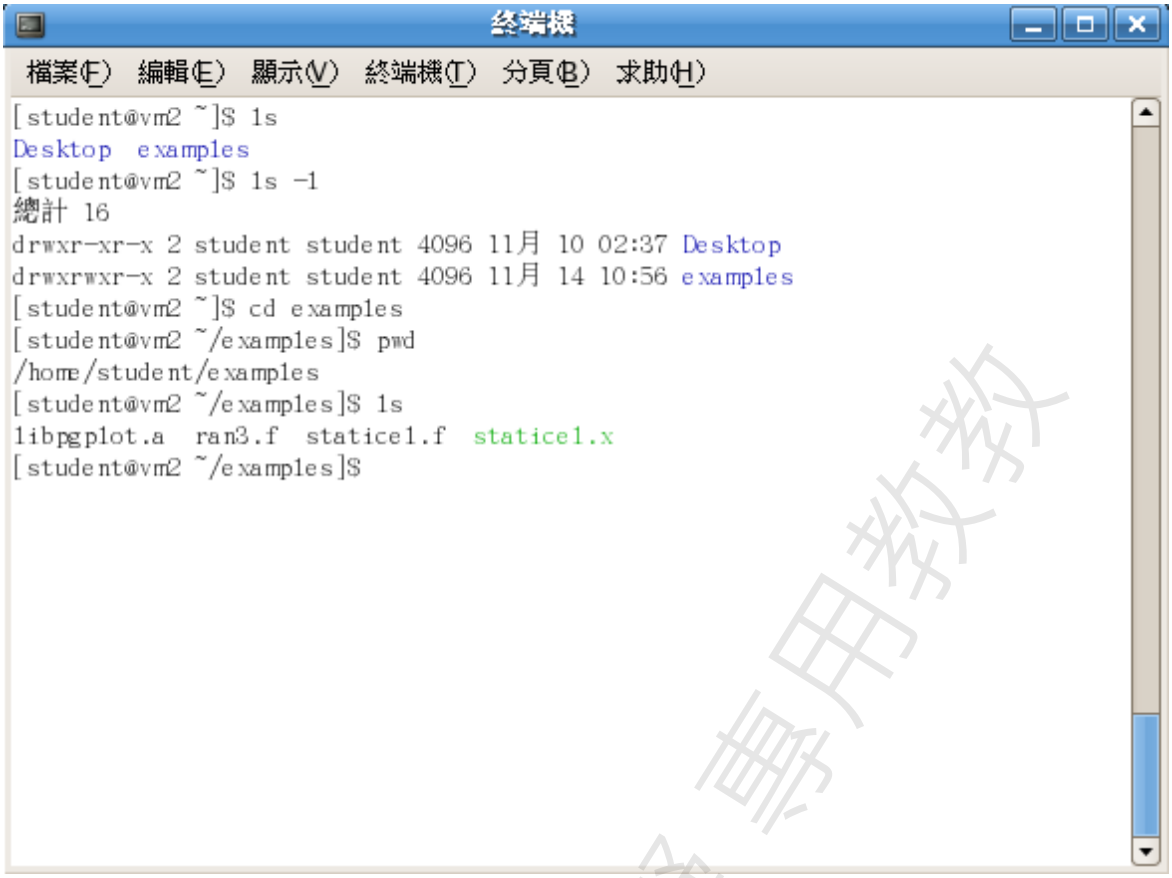
用 UNIX 指令 "ls" 來查看目前目錄下的檔案

`ls`（列出目前目錄下的檔案或子目錄名稱）

`ls -l`（以較長之訊息， 列出目前目錄下的檔案或子目錄名稱，注意 `ls` 與 `-l` 之間有空格）

pwd（印出目前工作目錄 present Working directory）

`cd "目錄名稱"`（變更目錄位置，進入到 "目錄名稱" 的目錄之內）



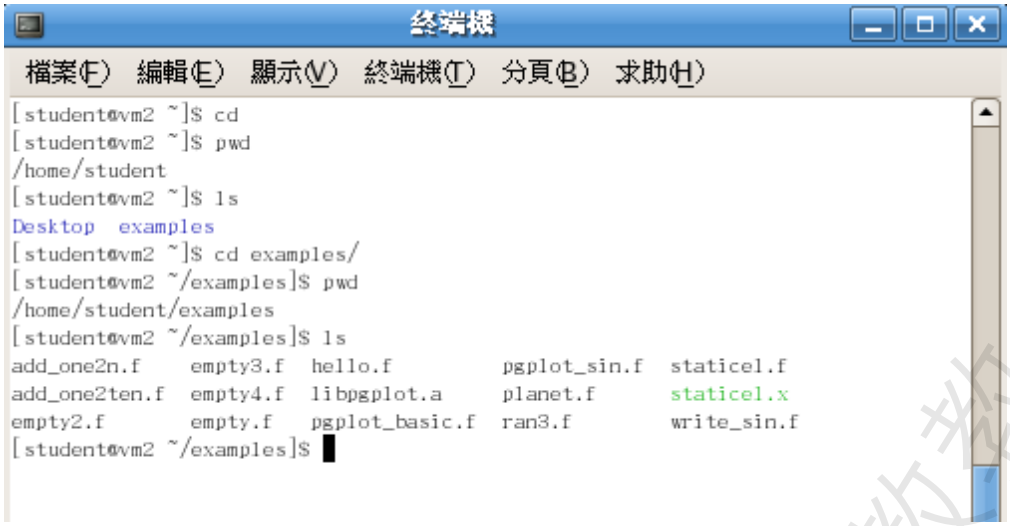
跑一下電腦模擬範例程式的可執行檔

在終端機視窗下，先打 cd 確實回到使用者 home 目錄，下 ls 可見 Desktop 與 examples 兩個目錄。打 cd examples 進入 examples 目錄，可用 pwd 印出目前目錄以供確認，然後打 ls 看這個目錄下有什麼檔案

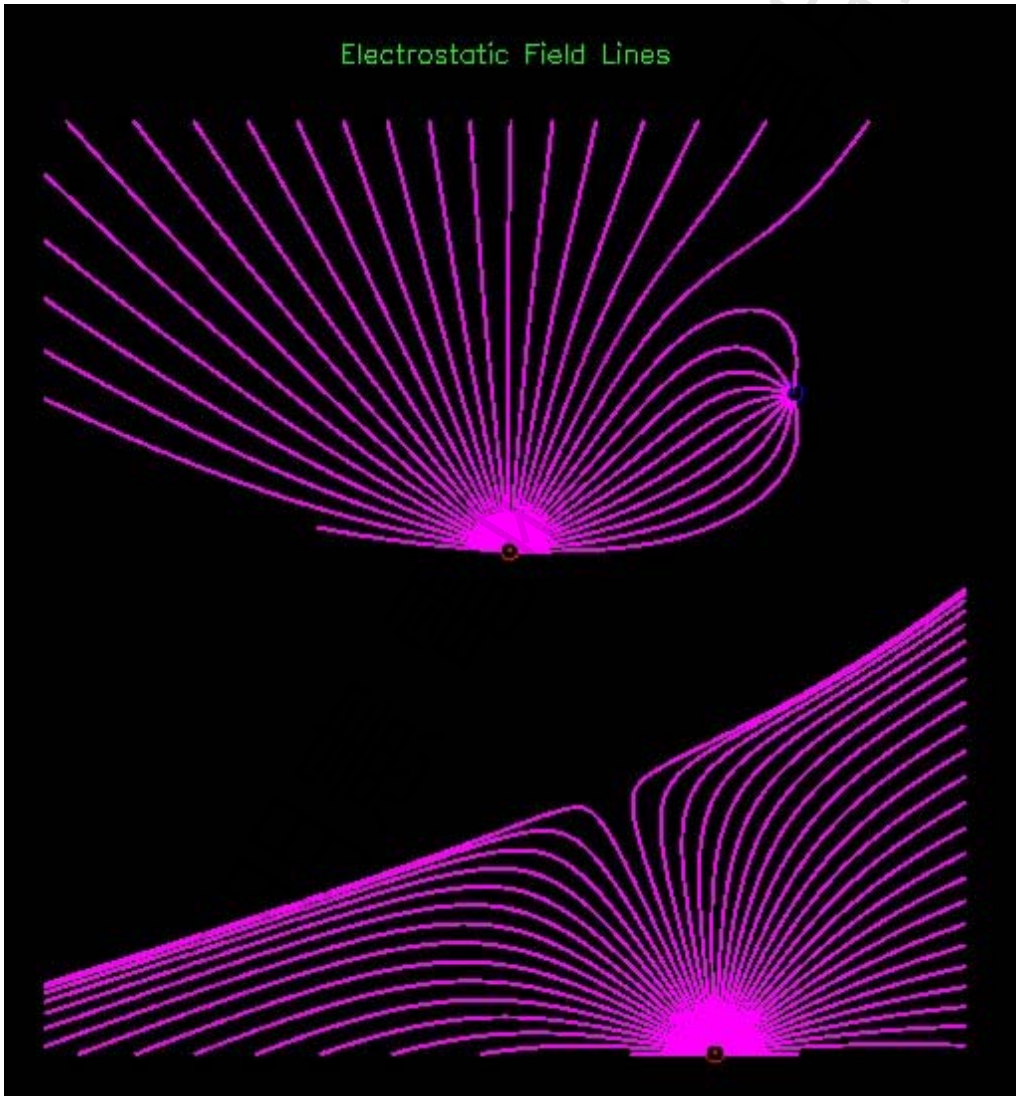
如果你使用的是 Fedora 11，而在使用者目錄之下沒有 staticel.x 檔案的話，請用瀏覽器（在最上一排那個有地球與滑鼠的圖示）來下載此一連結 `fieldline.f` `ran3.f`（按滑鼠右鍵另存新檔，預設的存放位置在 "下載" 目錄下。你如果不會用 Fedora 11 的中文輸入，還是可以用滑鼠右鍵滑動來標亮拷貝文字以及按滑鼠中鍵貼上文字。）再打

```
gfortran fieldline.f ran3.f -L /usr/local/pgplot -lpgplot -lX11
```

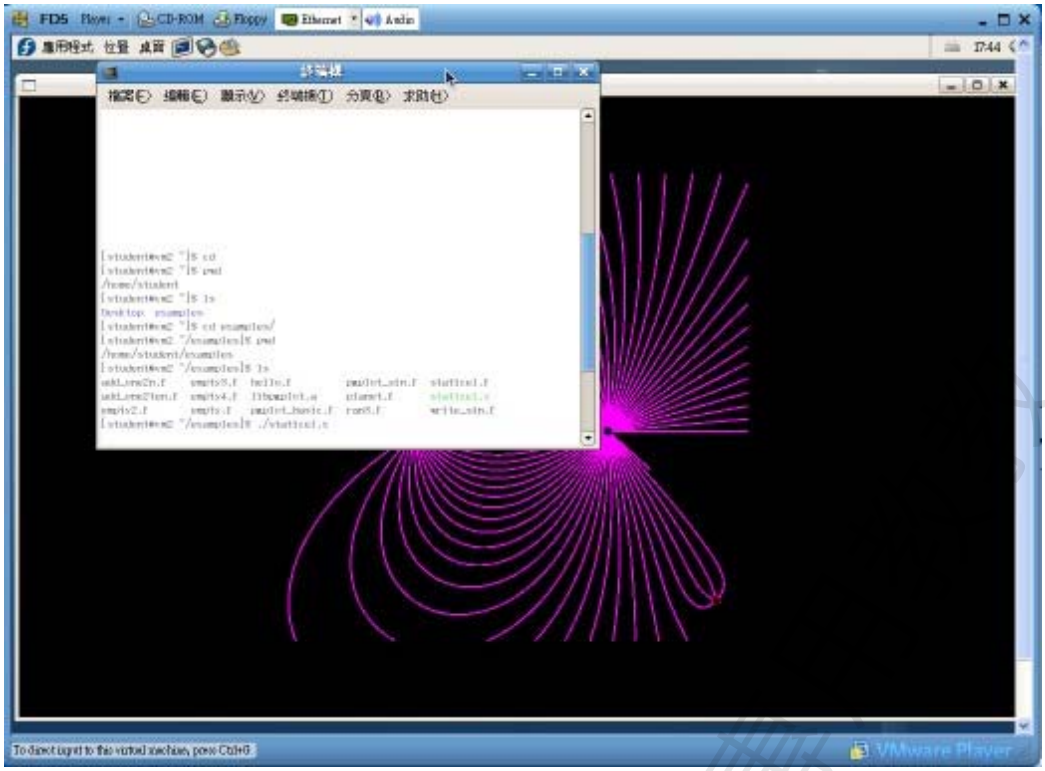
然後打 `./a.out <Enter>`



你會看到 `staticel.x` 這個檔案（綠色顯示它是一個執行檔，但這種顏色區分並非所有 UNIX 終端機皆通用），打 `./staticel.x` 並按 `Enter` 鍵，電力線模擬就會開始



把電力線展示圖關掉的方法，是先點選 terminal 上邊框使其回到最上層顯示，如下



再同時按下 `Ctrl` 與 `c` 把圖形展示程式 `staticel.x` 殺掉。

用 `vi` 分別開啓幾個範例檔案來檢視其內容，一次看一頁，並學習不更動而安全退出之方法

（當場講解與練習）

進入 vi 與退出 vi 的方法：進入是打 vi "檔名"， 檔案存在就顯示，不存在就新建；要退出 vi，是按 ESC 跳脫出打字機模式而入編輯命令模式（會再詳細講），打冒號與 q 如":q"，可不修改退出。若瀏覽過程中有不慎打入編輯指令，則在 ":q" 時 vi 會提醒你尚未貯存，則你必須打 "q!" 強制放棄修改退出。

若檔案是什麼字元都沒有，則螢幕左列由 ~ 佔滿，代表那裏事實上是沒有字元的。

按 Ctrl 與 f 看下一頁、Ctrl 與 b 看上一頁。

關於 vi 的進一步資訊請見我另一個網頁

以下來看幾個 Fortran (Fortran 77) 程式：

在 UNIX (Linux) 下的 Fortran 程式，一定要命名成 ####.f 這種延伸檔名是 .f 的名稱，編譯器才會接受。你可以用 vi 編輯器打 `vi test.f <Enter>` 來練習輸入以下的範例程式並編譯。

若要進一步知道 Fortran 的語法，可參見我所寫的 " Fortran 兩小時快速入門"，連結在這裏。

最簡單的，是定義了程式，但什麼也設做（注意從第七格開始寫指令，第六格要保留給延伸符號用），像下而這個範例：（又，Fortran 語言中大小寫是不分的）（另外，多加入些空行在其中也沒有影響）

```
    program empty

    end
```

一樣是什麼事情都不做，多寫了個註解。任一行的第一格打 c（即 comment 之意），則該行整個當作註解，完今與程式指令無關

```
    program empty
c This program does nothing, and this is a comment line.
    end
```

很多程式書的入門介紹都是教人家寫一個印出 "Hello, world." 的話到螢幕上，在 Fortran 的寫法是像下例這樣：

```
    program hello_world

    write(*,*) 'Hello, world.'

    end
```

這表示 Fortran 是用 `write` 來字東西，而後面的 `(*,*)` 代表 "（ 輸出到預設裝置，以預設格式爲之 ）"。

再來看一個能作計算 1 + 2 + 3 + ... + 10 的程式，注意使用變數要宣告（宣告其爲實數 `real` 或整數 `integer`），並且 `a = b + c` 在程式中代表把 b 加 c 的結果放入 a，而不是數學中之恆等式那種意思（要點的地方已標了藍色），

```
    program add_from_one_to_ten
c This is a program that will do 1+2+3...+10
c and print the result to screen
    integer i, sum
    write (*,*) 'Adding from one to ten ...'
    write (*,*)
    sum = 0
    do i =1, 10
        sum = sum + i
    end do
    write (*,*) 'The total is :', sum
    end
```

我們甚至可以寫一個可以先請使用者輸入 n，再進行從 1 連加到 n 的程式，重點處我們用藍色標出，如下：

```
    program add_from_one_to_n
c This is a program that will do 1+2+3...+n for a given n value.
```

```
      integer i, sum, n
      write (*,*) 'Please type in an integer value n for
```

- doing 1+2+3...+n :'

```
      read (*,*) n
      write (*,*) 'Adding from one to ten ...'
      write (*,*)
      sum = 0
      do i =1, n
          sum = sum + i
      end do
      write (*,*) 'The total is :', sum
      end
```

看完了能做計算的程式，現在來看一下能畫圖的 Fortran 程式，Fortran 本身並未內建繪圖指令，我們是藉由呼叫（Fortran 指令 `call`）pgplot 這個免費的自由軟體來達成

```
      program pg_basic
c This program demonstrate basic pgplot functions (routines).
      real y(100), x(100), delta_x, point_y(5), point_x(5)
      integer i, pgopen
      if (pgopen('?').le.0) stop
      delta_x = 5.0/100.0
      do i=1, 100
          x(i) = (i-1)*delta_x
          y(i) = x(i)**2
      end do
      do i=1, 5
          point_x(i) = i
          point_y(i) = i**2
      end do
      call pgenv(0.0, 10.0, 0.0, 20.0, 0, 1)
      call pglab('x','y', 'y=x**2')
      call pgline(100, x, y)
```

```
call pgpt(5, point_x, point_y, 9)

call pgclos

end
```

這個 pgplot 程式被編譯成可執行檔之後，執行時會問繪圖裝置，回答 /xwin 即可（注意含斜線）。

有關於 pgplot 的進一步資料，請看我另一個網頁。

本段各範例程式的連結點是 這裏，可自行用瀏覽器下載至工作目錄下。

試用系統內建之 fortran compiler 來產生那幾個程式原始碼的可執行檔，並試著執行它們

（當場講解與練習）

若是**一般計算的程式**，以 hello.f 爲例，只需打（我們以使用 gfortran 這個 fortran compiler 爲例，在別的系統你有可能會用到的是 f77 或 g77）：

```
gfortran hello.f <Enter>
```

以上作法其可執行檔會一律叫做 a.out（要執行時，打 `./a.out <Enter>`）。或者是你也可以打：

```
gfortran -o hello.x hello.f <Enter>
```

則其可執行檔會依你要求被命名爲 `hello.x`。（要執行時，打 `./hello.x <Enter>`）

需要用到 pgplot 繪圖的副程式庫（即目錄下的 libpgplot.a）時，則略爲複雜一點，例如你的程式若是叫做 pg_basic.f ，就要打（注意以下文字要打在同一行，不能用 <Enter> 斷開）：

```
gfortran -o pg_basic.x pg_basic.f -L /usr/local/pgplot
-lpgplot -lX11
```

其中 `-lpgplot` 是告訴編譯器要額外引入 `libpgplot.a` 那個繪圖副程式庫檔案裏的相關副程式的，而 `-L /usr/local/pgplot` 則是說明該檔案在那一個目錄（即資料夾）下，至於`-lX11` 則是告訴 fortran 編譯器要引入 X－視窗繪圖的動作的基本程式庫 X11（其實際檔名叫 libX11.a），因此兩者都是需要的。

在 UNIX 之下使用 Fortran 編譯器的相關 進一步資料 ，可自行參考。

用 vi 寫一、兩個最簡單的程式，順便學用 vi 的最基本指令

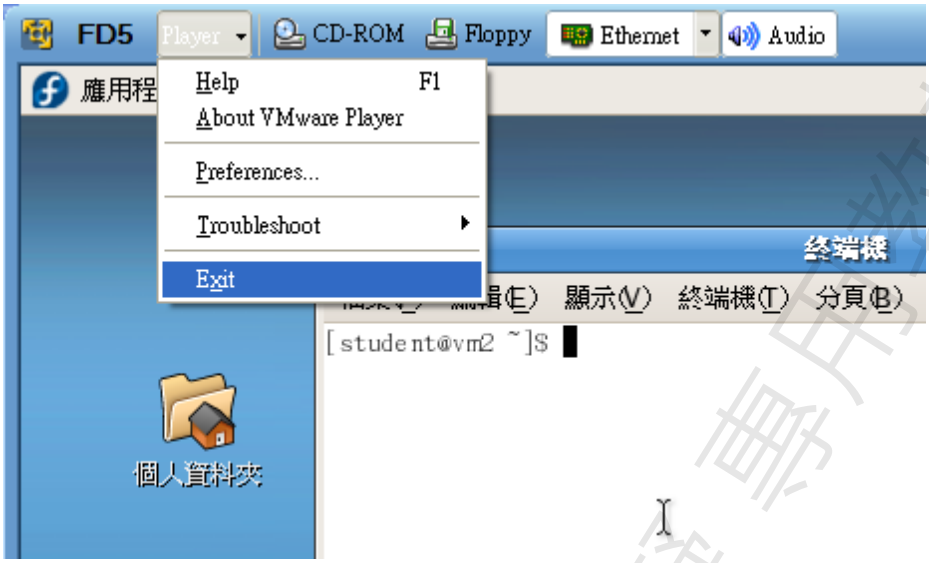
（當場講解與練習）

關於 vi 的進一步資訊請見我另一個網頁，fortran 兩小時快速入門則看這裏。

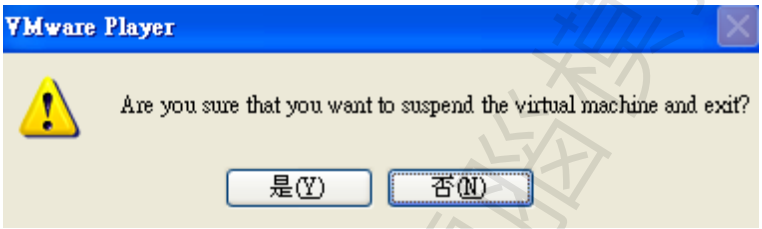
登出系統及關機或虛擬機器休眠之方法

休眠

最方便的作法，是無論工作到那個階段，就直接將虛擬機器 **直接休眠**，從 VMware Player 左上角 Player 下拉選單中按 Exit



會要求確認動作



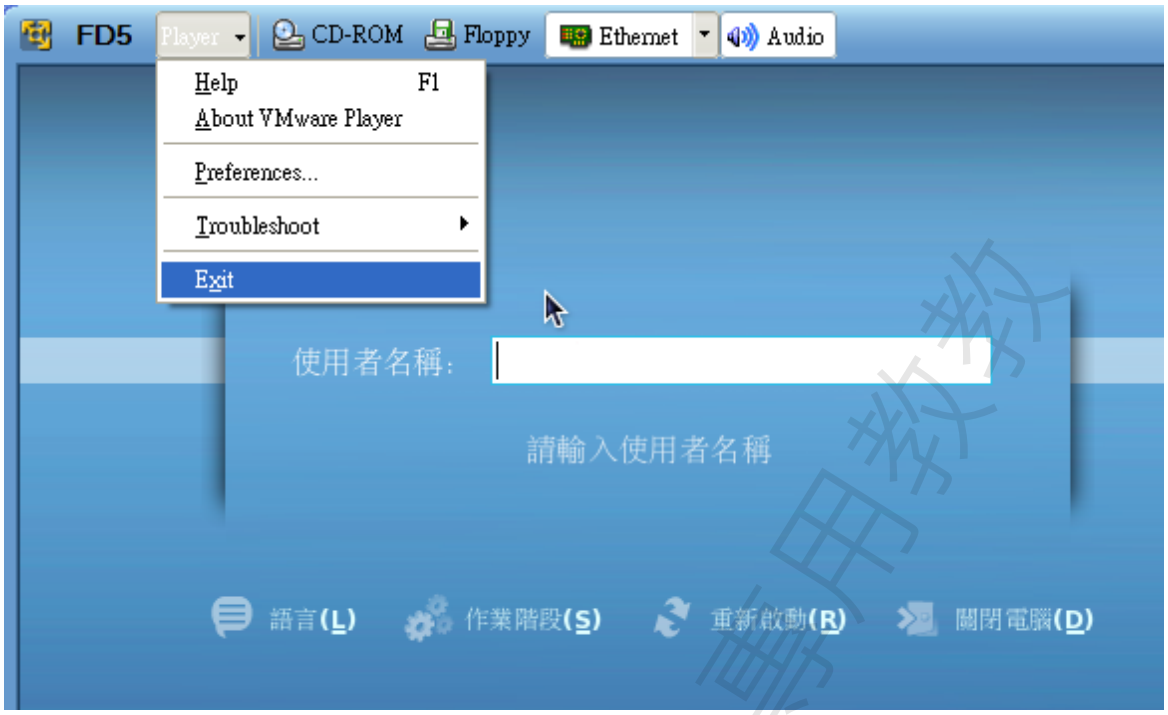
虛擬機器就會開始休眠，注意需要給它一點時間，VMware Player 才能完成並退出。

下次再用 VMware Player 開啓虛擬機器時，就會從休眠前的畫面繼續運作下去，因此這是最方便的作法。

登出

要從上排圖示中的桌面打開選單，並選登出按下

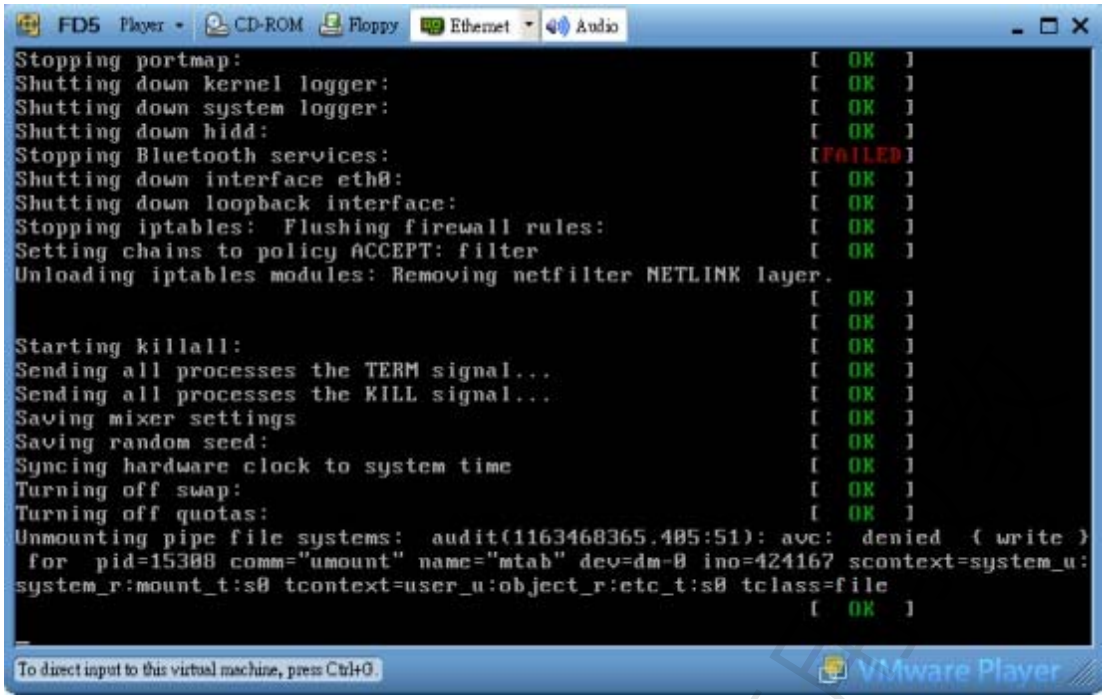
這樣就會回到登入畫面，你也可以選擇休眠，下次再啓動至少省下開機過程所花的時間



關機

這裏指的是虛擬機器關機，你可以在登出後關機，或是在 X-window 工作到一段落之後就從上排圖示的桌面選單中按關機。（關機在真實伺服器或多人使用環境才會與休眠的效果比較不同，因此在此並不常用。）

要求確認的對話框會出現，確認關機後，系統會逐步關掉其在背景中運作的程式而有逐條的訊息出現



待關機完成，整個 VMware Player 也會隨即退出。

如果還有時間，請大家進行額外的 Fortran 程式撰寫練習／複習：

電腦計算部分

a. 整數從 1 加到 10 並且把答案在螢幕印出

偷看一下參考範例程式

b. 把上題改爲向使用者詢問從 1 加到 N 的 N 值，並且把答案在螢幕印出

偷看一下參考範例程式

c. 同上，但把整數從 1 加到 N 的結果寫到檔案

偷看一下參考範例程式

d. 把上述程式的連加部分寫成一個副程式

偷看一下參考範例程式

電腦繪圖（pgplot）部分

e. 呼叫 pgplot 副程式 pgpt 來畫幾個點

偷看一下參考範例程式

f. 呼叫 pgplot 副程式 pgline 來畫連筆曲/折線

偷看一下參考範例程式

g.呼叫 pgplot 副程式 pgcirc 來畫一個圓

偷看一下參考範例程式

h.呼叫 pgplot 副程式 pgsci 來作動畫

偷看一下參考範例程式

落體的運動

一、電腦模擬實驗：自由落體問題

問題：想知道炮彈打到那裏

物理量：軌跡隨時間的變化 x(t), y(t)

物理定律（數學模型）：**f** = m **a** ，在此 **f** 純粹來自於重力（**粗體字**代表是向量）

粒子在地表附近所受的重力是 g （g = GmM/R），g = 9.8 牛頓/米

f = m**a** = m (a_x $\mathbf{e}_x$ + a_y $\mathbf{e}_y$) ，其中 $a_x = dv_x/dt$ 、 $a_y = dv_y/dt$

只有重力時，**f** = f_y $\mathbf{e}_y$ = g m $\mathbf{e}_y$ ，也就是說 f_y = g m

演算法：

利用 Euler 演算法：

位置 $x_{n+1} = x_n + v_{x,n}\,Dt$ ； $y_{n+1} = y_n + v_{y,n}\,Dt$

速度 $v_{x,n+1} = v_{x,n} + a_{x,n}\,Dt$ ； $v_{y,n+1} = v_{y,n} + a_{y,n}\,Dt$

進階補充：對計算穩定性及精確度，有更好的選擇

Euler-Richarson 演算法， 先求出

$a_{x,n} = f_x(r_n, v_n, t)/m$

$v_{x,mid} = v_{x,n} + (1/2)\,a_{x,n}\,Dt$

$x_{mid} = x_n + (1/2)\,v_{x,n}\,Dt$

$a_{x,mid} = f_x(\,x_{mid},\, v_{x,mid},\, t+(1/2)Dt\,) / m$

然後才做

$v_{x,n+1} = v_{x,n} + a_{x,mid}\,Dt$

$x_{n+1} = x_n + v_{x,mid}\,Dt$

註：爲什麼 Euler-Richarson 演算法 比 Euler 的好，可參考 Gould 課本上的推導： p1、p2

程式流程：

(0) 開始，宣告變數

(1) 問質量、初速度（含初速率與角度）

(2) 建立位置 x 與 y、速度 v_x 與 v_y 的初始值

(3) 用演算法求下一個小時段 Dt 後的位置與速度

(4) 畫出粒子位置並檢查高度是否已達地面，未達地面則回到 (3)

(5) 問要不要再射一次，若要則回到 (1)

(6) 結束

你等不及了嗎？先偷玩一下老師已經做好的可執行檔 `cannon.x`

你做不出來嗎？先偷看一下老師已經寫好的範圍程式 `cannon.f` `cannon.f.txt`

（非不得已請勿先看）

動動腦、動動手，自己再改造（或回家做）：

1. 畫個大砲把炮管的角度表現出來
2. 以火藥包裝填數表現炮彈動能，轉換成初速率
3. 物體被炸破片的彈出（配合亂數產生器 ran3.f，使用方法）
4. 試寫一個煙火程式，會多段開花變色的

分析與討論：

想預測炮彈打到那裏，主要是基於那一條物理定律？

什麼樣的先決（初始）條件決定了炮彈的軌跡（和落點）？

同樣的火藥包充填數，怎麼打炮彈才飛得最遠？

二、電腦模擬實驗：受空氣阻力的落體

問題：在有空氣阻力的情況下，炮彈軌跡變得如何？

物理量：一樣是軌跡隨時間的變化 x(t), y(t)

物理定律（數學模型）：**f** = m **a**，受力 **f** 同時包含了重力及空氣阻力

空氣阻力非常複雜（因此飛機或高速火車的阻力設計需要靠風洞實驗或電腦模擬來進行），基本上它是與速率有關，速率越高阻力越大，靜止的物體則沒有空氣阻力。可見它是一個與速率有關的量，常見的簡化公式，有隨速率一次方呈正比的，也有隨兩次方呈正比的，方向是速度反向（負號）的

$F_d = - k_1 v$

$F_d' = - k_2 v^2$

由於有 x 與 y 兩個分量，假設速度與水平線的夾角爲 q ，即 q = $\tan^{-1}$(vy/vx) [Fortran 程式寫成 `theta = atan(vy/vx)`]，

$F_{d,x} = F_d \cos q$

$F_{d,y} = F_d \sin q$

（或者是，直接用 vx/sqrt(vx**2+vy**2) 及 vy/sqrt(vx**2+vy**2) 會比先求 theda = atan(vx/vy) 取角度才去求 cos(theda) 及 sin(theda) 更好）

因此，在有關力之部分的公式，

$f_x = F_d \cos q$

$f_y = - m g + F_d \sin q$

其他的部分，與上一個議題相同

新議題：如何求得室氣阻力係數 k？

利用終端速度，即垂直落下過程中重力與阻力達平衡，使合力爲零不再加速，而呈等速運動時的速度。

合力 = 0 = $- m g + F_d = - m g + k_1 v_y^2$

即 $k_1 = v_y^2 / (m g)$，此終端速度可藉由實驗量得

一個粉筆頭大小的石子其終端速度大約是每秒三十公尺。另外，一個重 0.254 克 、半徑 2.54 公分的保利龍球有以下的實測數據（引用 Physics Teacher **24**, 153 (1986) ）

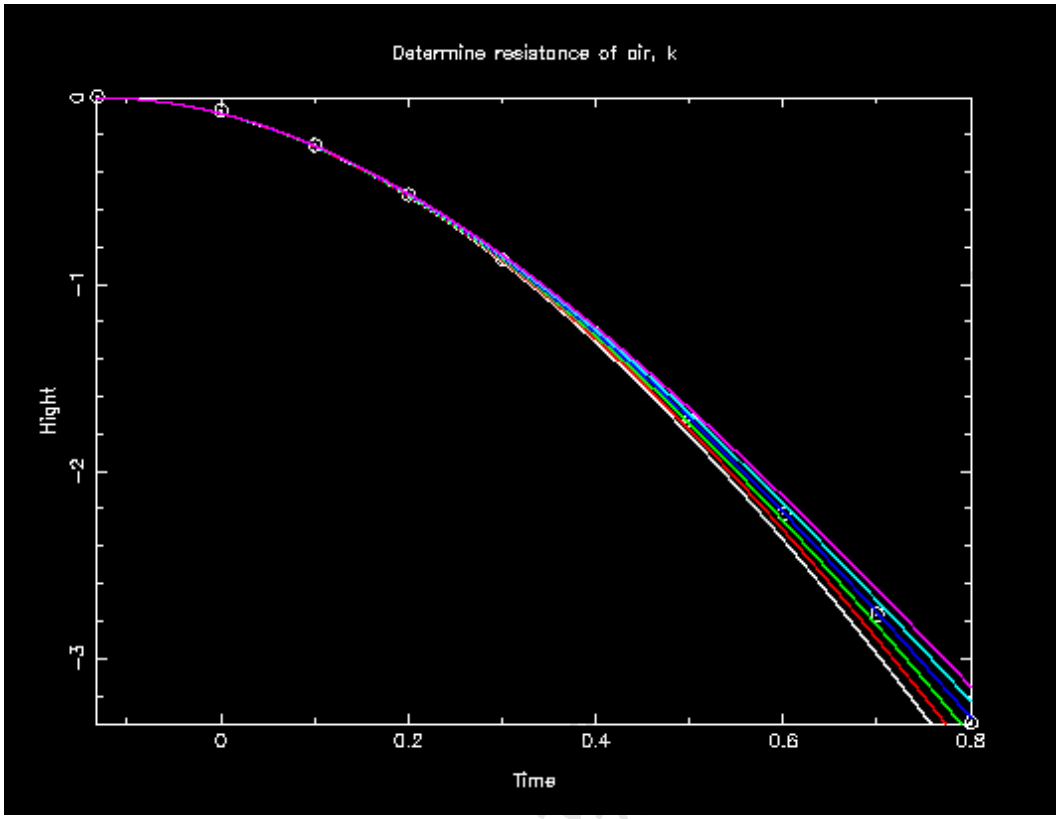
時刻（秒）	位置（公尺）
-0.132	0.0
0.0	0.075
0.1	0.260
0.2	0.525
0.3	0.870
0.4	1.27
0.5	1.73
0.6	2.23
0.7	2.77
0.8	3.35

想想看，我們要怎麼從上列的數據求得空氣阻力係數 k？

老師寫的範例程式：styrofoam_k.f.txt styrofoam_k.f styrofoam_k.x

上面表格數據的資料檔：styrofoam.txt

（儘量自己想，不會再參考）



演算法：

同樣採用 Euler 或 Euler-Richardson 演算法

程式設計：

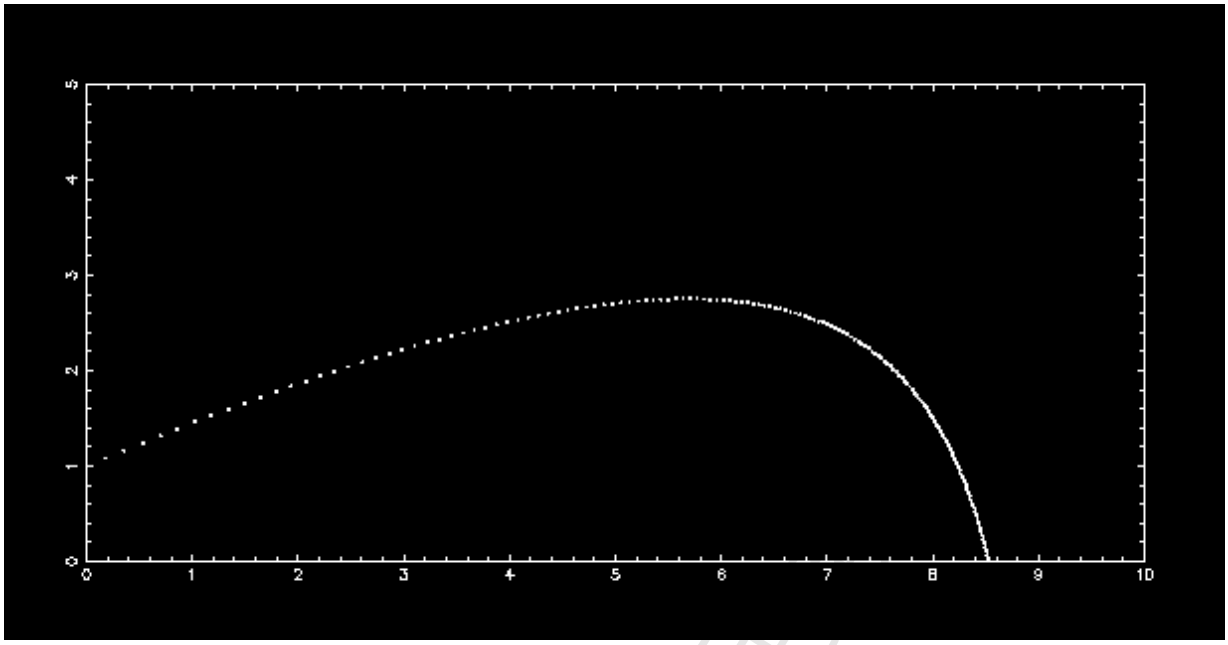
與上一個議題相同

參考範例程式

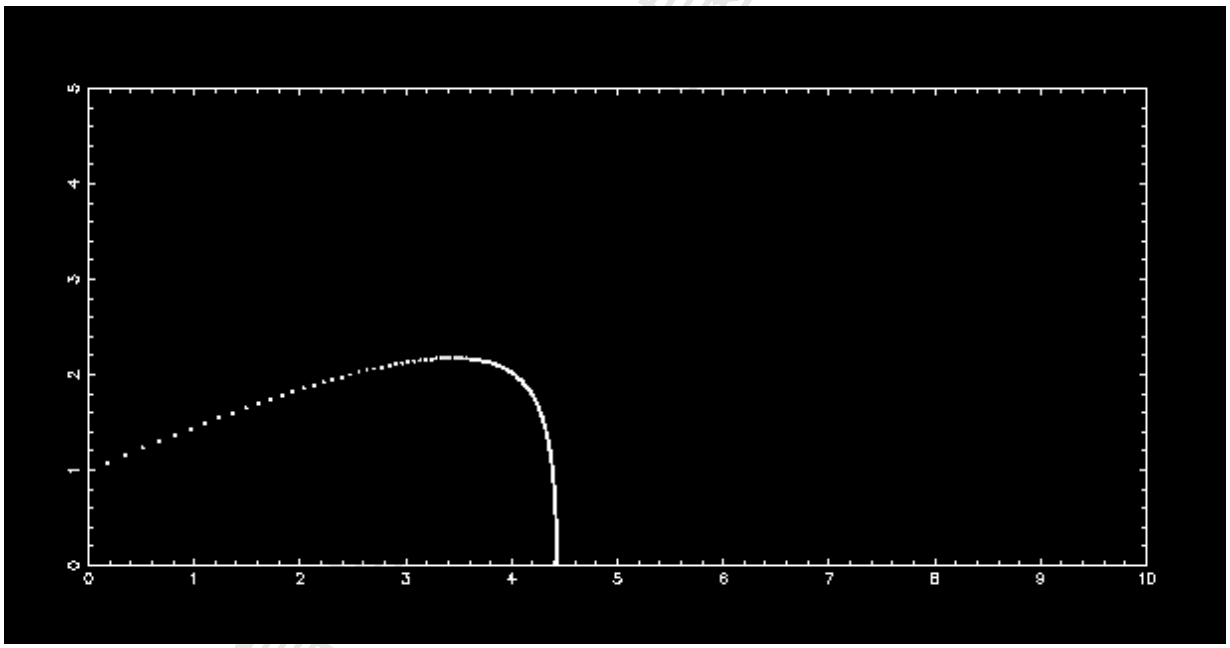
cannon_drift.f cannon_drift.f.txt cannon_drift.x

質量與空氣阻力係數對拋射體軌跡的影響

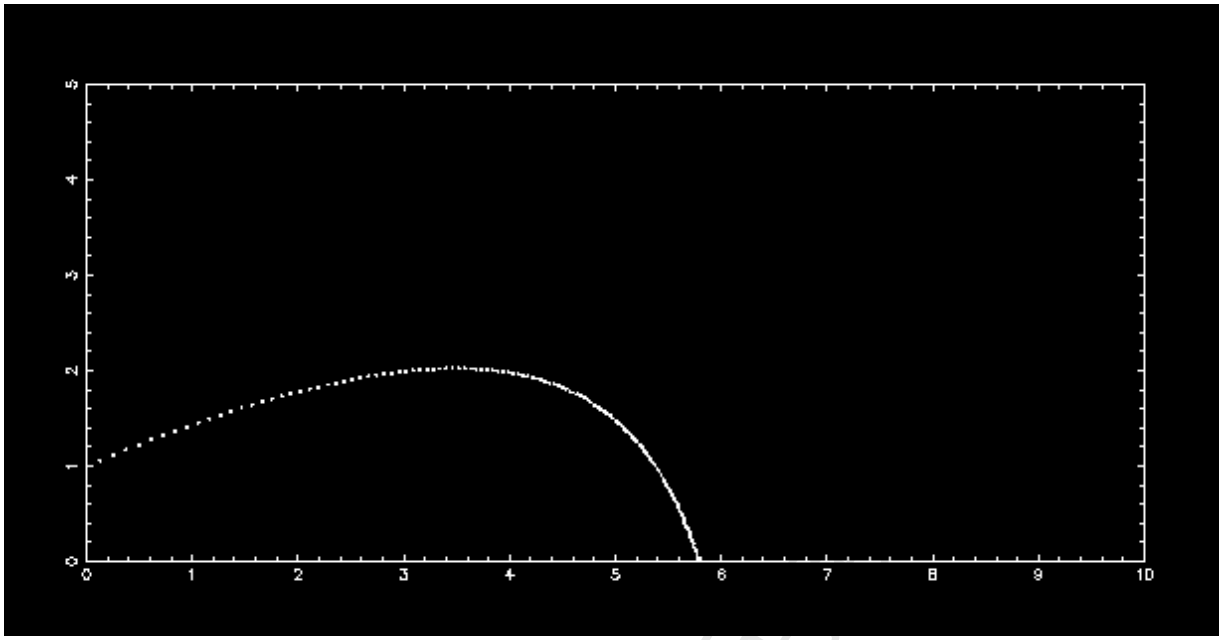
mass = 1.0, k = 2.0



mass = 0.5, k = 1.0



mass = 2.0, k = 2.0



分析與討論：

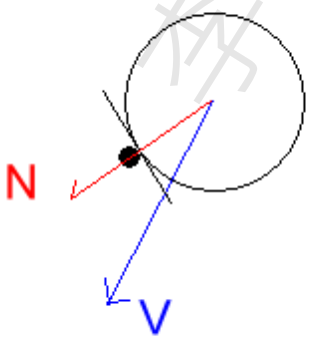
以同樣的力道向上拋一個小石子，在有空氣阻力與無室空氣阻力的兩種情況比較下，那一種會在空中停留較久？

那一種軌跡像羽毛球？那一種像鉛球？

在有空氣阻力的情況下，拋射物的角度要如何設定其距離才會最遠？

作業：彈珠台程式

彈珠與光滑釘子的交互作用模式



N = N_x $\mathbf{e}_x$ + N_y $\mathbf{e}_y$

| **N** | = 1

V dot **N** = V_x * N_x + V_y * N_y

碰撞後 V 平行於 N 上的分量因反彈而方向相反，與 N 垂直的分量則繼續維持，因此碰撞後

$\mathbf{V}_\perp$ = (-1)* (**V** dot **N**) **N**

$\mathbf{V}_{//}$ = **V** - (**V** dot **N**) **N** = **V** + $\mathbf{V}_\perp$

新的速度則爲 $\mathbf{V}_{//}$ + $\mathbf{V}_\perp$

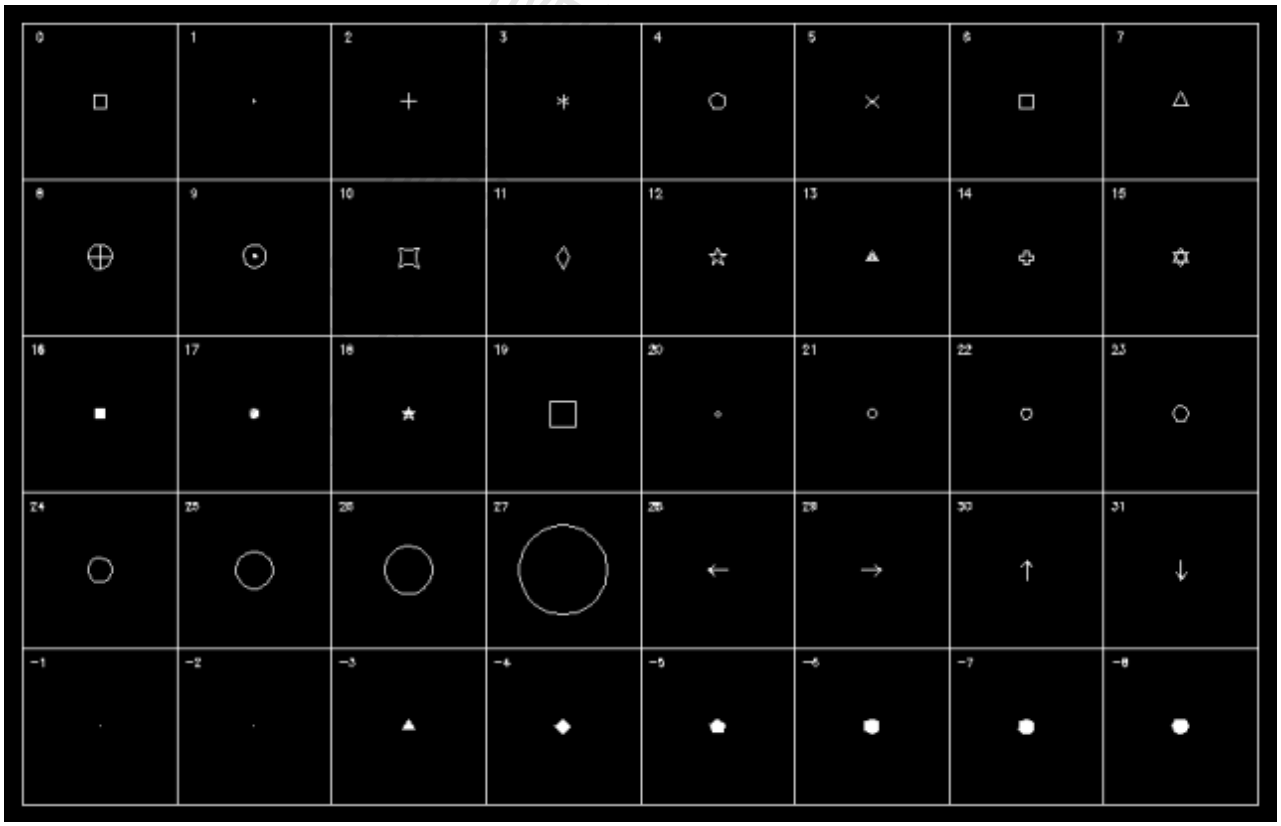
老師提供的的範例程式 `pinball.f` `pinball.f.txt` `pinball.x`

動手改造與討論：

將你的程式改爲一次釋放兩顆、三顆彈珠。

如果是對彈珠有摩擦力的釘子，則彈珠與之交互作用模式如何？

PGPLOT： PGPT (n,x,y,SYMBOL)



亂數產生器的用法

以 ran3.f 內含的 ran3(iseed) 函式爲例：

```
iseed = 7
x = ran3(iseed)
y = ran3(iseed)
```

準備任意一個整數值放在某個整數變數 iseed 的值內，依上述的方法每次使用 ran3( )，則 x 與 y 都會獲得一個 0.0 到 1.0 之間的數值，每次的數值雖然不一樣，但這些數值在 0 到 1 之間出現的機率是均等的。另外，直接把整數填入 ran3( ) 卻並不支援，如 `xx = ran3(7)`，會導致 "區段錯誤 (Sgementation Fault)" 。

試試看，寫一個丟骰子程式。例範：`dice.f` ，記得與 ran3.f 一起編譯，如 `gfortran diec.f ran3.f <Enter>`。

另外 gfortran 支援內建副程式 `random_number(`*實數變數*`)`，就不必另外用外部的亂數函式了，可參考另一骰子程式範例 `dice_gfortran.f`。重設亂數種子的副程式是 `random_seed` 細節請見： http://www.nsc.liu.se/~boein/f77to90/a5.html#section21c

進階閱讀：

Gould and Tobochnik, An Introduction to Computer Simulation Methods -- Applications to Physical Systems, Addison Wesley (1996) Chapter 2, Chapter 3

電力線與電磁波

靜電場的電力線

電力是一種超距力，兩個帶電體在空間中並不接觸就能互相影響。法拉第想像空間中佈滿電力線，源自正電荷且終於負電荷，線越密集的地方電場越強，電力線永不相交。

在有電荷 q_i {i=1,N} 分佈的空間中，其處處的電場向量 **E**(**r**) 公式是

$$\mathbf{E}(\mathbf{r}) = K \sum_{i=1}^{N} q_i (\mathbf{r} - \mathbf{r}_i) / |(\mathbf{r}-\mathbf{r}_i)|^3$$

利用電腦模擬的方法，我們可以看到電力線的分佈。Gould 一書提出一套在平面作圖上相當有效率的方法，在此我們拿來作觀摩介紹，其步驟是：

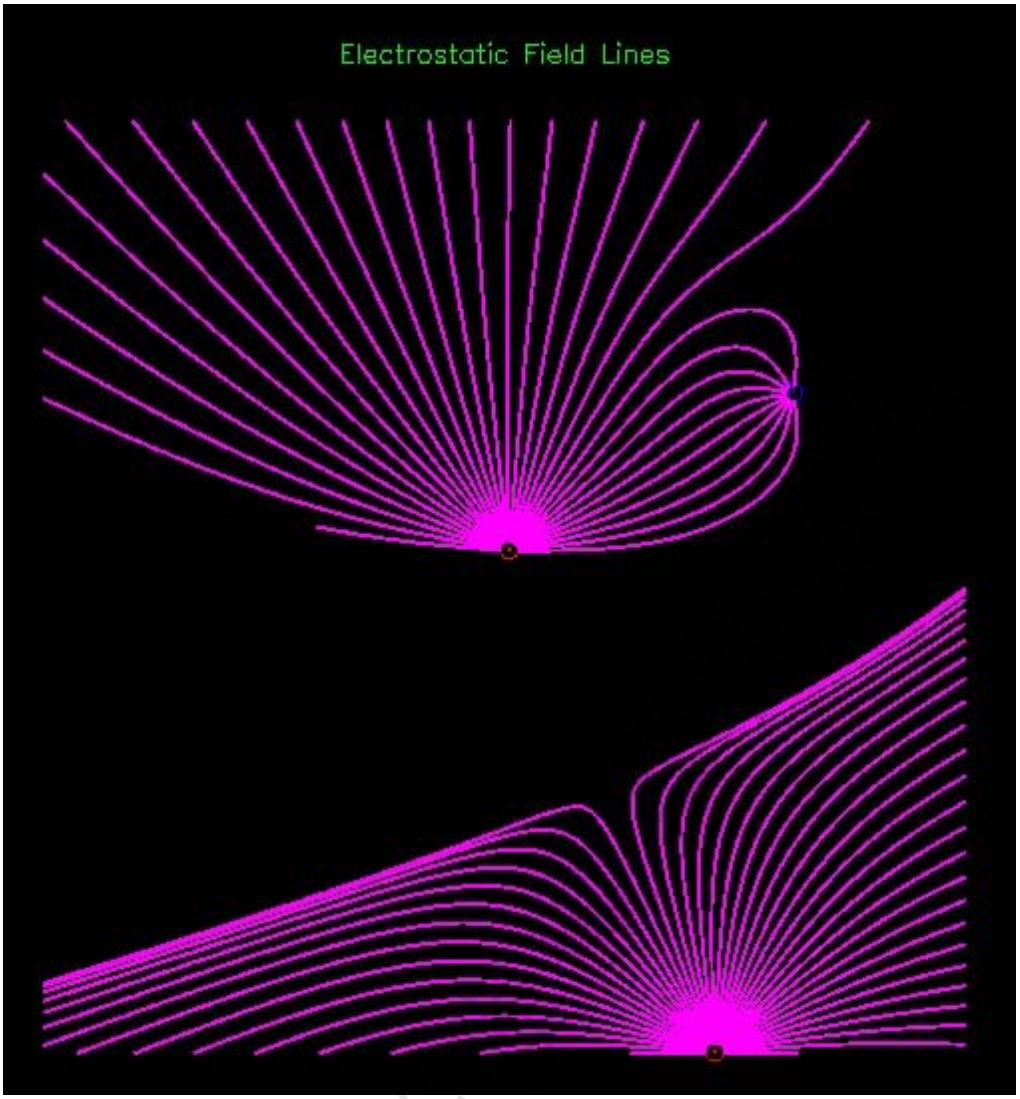
(1) 以每個電荷爲中心，依其電荷大小，決定其應射出的線條數，安排成等分角度幅射狀射出。

(2) 每條電力線自電荷出發後，循著電場方向一小步一小步走出一條線，直到抵達另一電荷，或者是越行越遠以至於電場越弱而小於某一個值，我們可認爲已無電場而停止繼續畫電力線。

(3) 爲了提昇速度，在電場較小或遠離電荷的空間位置，用百倍於小步幅的大幅度畫線。（此一技巧純粹是效率考量，並非必須之措施。）

(4) 我們替這個觀摩程式多加了以亂數產生器來隨機地設定電量與位置，來作自動展示之用。

觀摩程式：`fieldline.f` `fieldline.f.txt` `ran3.f` `ran3.f.txt`



Gould 原課文參考頁：[p1](#)、[p2](#)

非線性動力學以及混沌現象模擬

Nonlinear Dynamical System

一般按英語字面意義翻譯作 "非線性動力學系統"，也很常見。其實在這裏的 dynamical 意指系統的狀態時時可以改變，其用法比較像是 DRAM (Dynamical Random Access Memory) 動態隨機存取記憶體，Dynamical Memory Allocation 動態記憶體配置，是 "動態" 的意思，因此 Nonlinear Dynamical System 應稱作是 "非線性動態系統" 才更不易引起誤解。當然，在物理學的範疇中有很多的力學系統也是動態系統，因此針對那些系統以 "非線性動力學系統" 稱之也沒錯，只是這樣對於其他非機械或非力學的系統，如動態的電路訊號變化或動態的人口變化，這名稱就有一點怪怪的了。

什麼是複雜？什麼是簡單？

複雜與簡單雖然只是兩個不具精確物理意義的形容詞，但在這裏至少有一個大家應該都能接受的區分，就是一個現象有沒有週期性。有週期性的現象，預測起來比較簡單，像日昇日落、春夏秋冬；而沒有週期性的現象，諸如氣象變化、股市脹跌。

一個現象會複雜，會令我們無法預測，難到都是因爲系統構成的元件很多而個自之間的交互作用又複雜，像封閉容器中所有氣體分子的動向那樣嗎？並不盡然，一個只具有相當小自由度的，服從一個簡單但卻是非線性的公式來變化(演進)，也可以產生複雜到無法預測其週期的情況，在物理的現象中我們稱爲是混沌現象。

註：混沌一辭很早就有，但與現今用於科學領域上的不一樣，像是小孩子讀的《幼學瓊林》中有 "混沌初開，乾坤始奠，氣之輕清，上浮者爲天；氣之重濁，下凝者爲地"。文言文中的混沌並不是物理名詞，它主要是要描述宇宙的 "無結構性"。另有人引述爲“混沌初開，乾坤始奠，輕清者上浮而爲天，重濁者下凝而爲地”

現在翻譯作混沌（Chaos）的英文原意是混亂，氣象學家勞倫斯用混沌理論，是用來描述僅有些微之差的初始條件就會導致後續的系統狀態有極大的不同，就像是所謂的蝴蝶效應（請見維基百科）那樣。

有些東西的本質本來就是隨機性的，如擲一個骰子會不會出現六點這件事情。並不因爲初始條件的些微不同而有異，就不是在這裏所謂的複雜了。

一個簡單的非線性動態系統：一維 logistic 映射

生物族群個體總數 population 與世代的關係

$$P_{n+1} = P_n (a + bP_n)$$

提出 a 係數後變爲 $a P_n [1 + (b/a) P_n]$，再變換變數 $x = (b/a)P_n$ 後則原式簡化成爲 $(a^2/b) x_n (1+ x_n)$，若把 a^2/b 再改爲 4r 則有 logistic map

Logistic map 的定義

一連串 x | -> x' 、 x' | -> x'' 的映射，每次得到的結果又作爲下一輪的術輸入值，其中第 n 次與第 n+1 次具有以下關係：

$$x_{n+1} := 4r\, x_n\, (1-x_n)$$

提示：若是寫成程式，就只須要寫一行 `x = 4 * r * x * (1-x)` 。

非線性的物理意義

讓我們先回想一下化學反應速率與反應物種濃度的問題。假設速率決定步驟是 A + B → AB，則正向反應速率正比於個別反應物濃度 [A] 與 [B] 的相乘 [A][B]，又，如果決定步驟是 A + B + B → AB_2，則正向反應速率正比於 $[A][B]^2$。 我們都可以理解這是因爲要形成產物，（以後者爲例）需要三個東西碰在一起，因此有 [A]、[B]、[B] 密度相乘來反映出機率。

假設食物充足等環境因素使然，繁殖率與前一世代的數量呈正比，則式子中會有 x 一次方項；另外，昆蟲疾病的傳染，會經由兩個個體的接觸，如此有了$- x^2$ 項，如此就有了

$$x_{n+1} = 4r\, x_n\, (\, 1 - x_n\,)$$

做做看，如果是 x (a - x) ，其中 a 是不等於一的某一個同定常數（也就是說上式的 x 的 $- x^2$ 並不一定是要相同比例），如此更合理，則結果會如何？（這樣的壞處是沒有歸一化，x 的值的範圍不被局限在 0 到 1 之間。）說明：以 $a\, x + b\, x^2$ 爲例，可轉化爲 a (1 + c x) x ，(其中 c = b / a) ，再進一步變成 a/c (1 + cx) cx，最後把 cx 令爲最後的 x ，就有 x(1+x) 的形式了。

我們也可以從另一個角度來了解非線性項，若 x 是夠小的值，則現象可次包含高次的效應，而表示成汲級數展開（像泰勒展開式那樣），因此二次或更高次的效應之出現是相當自然的。

程式習作：畫出 Logistic Map 之 x 對 n 的作圖

蟲口隨世代作圖

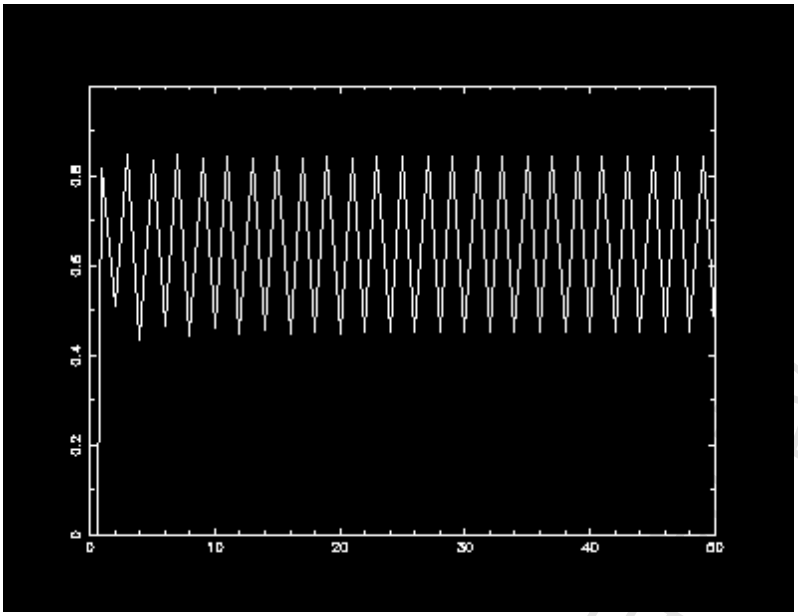
觀察到：

(1) 進入了叫固定點 (fixed point) 的穩定狀態（與初始蟲口值無關）

(2) 有些（較大的）r 導致會有兩個固定點

logistic_map.f

logistic_map.x

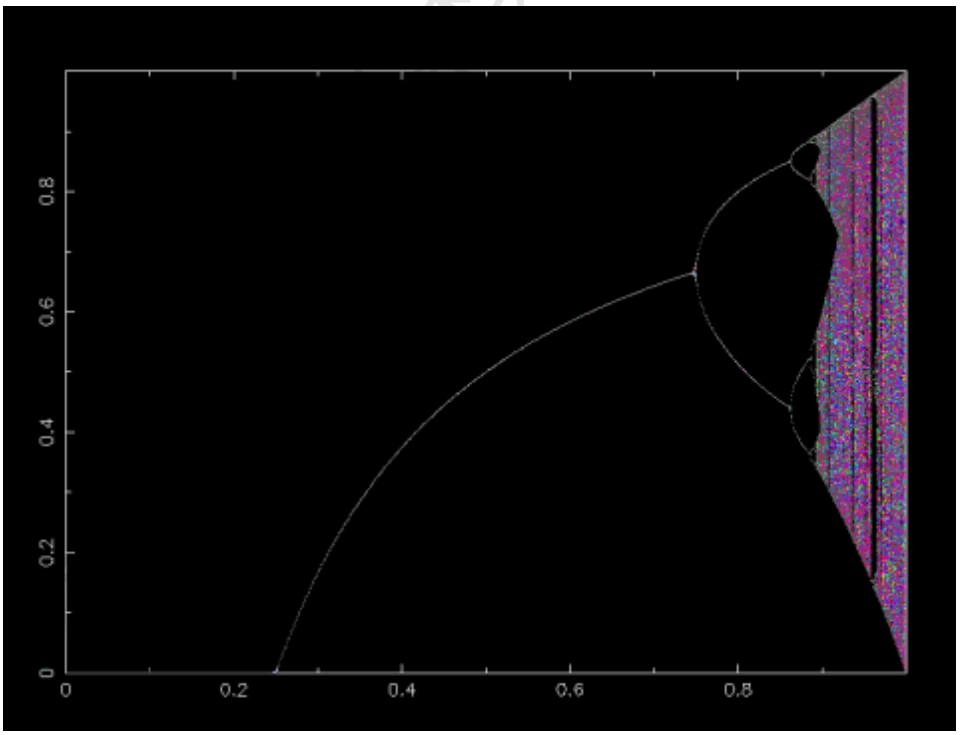


從幾何作圖的角度來看固定點分佈與數量上（隨 r）的變化

透過一系列作圖可呈現。

bifurcation.f

bifurcation.x



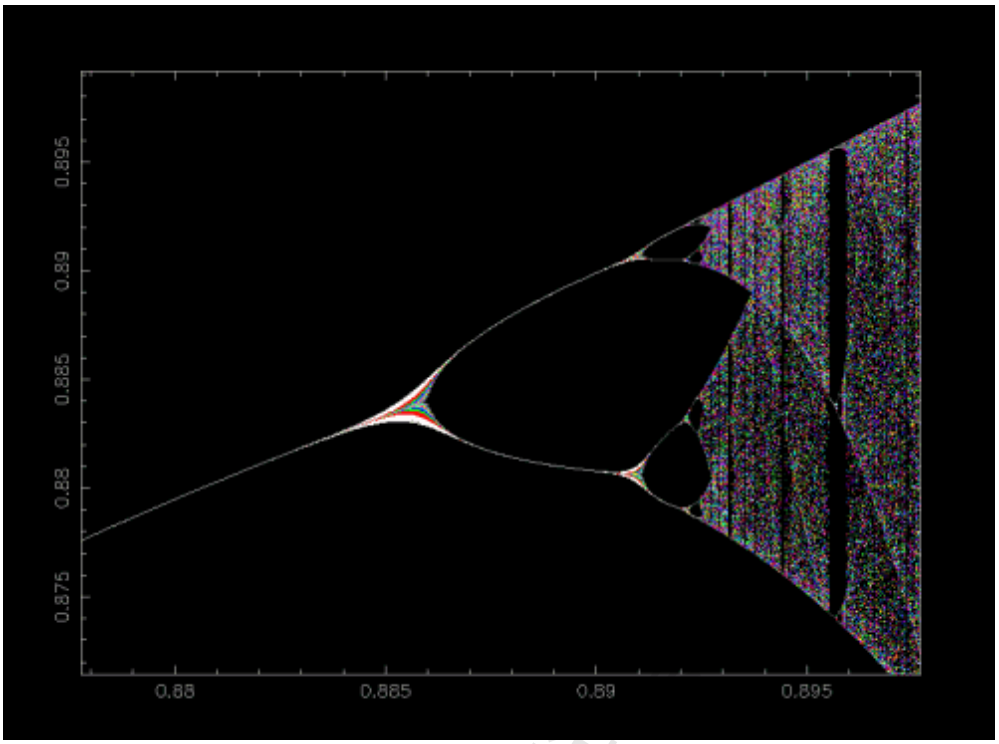
分叉圖的結構：碎形

放大某一小範圍的 r 來看，我們會見到重覆出現的類似結構。

思考：這是爲什麼？

bif_enlarge.f

bif_enlarge.x



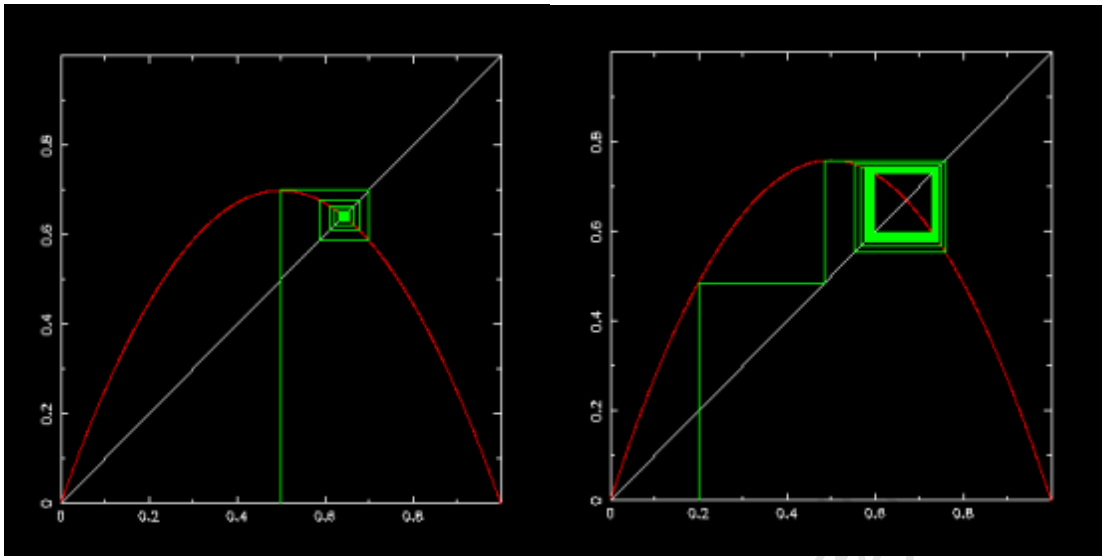
爲什麼分叉會發生？從函數的圖形來了解

Logistic Map 如果寫成一個實數形式，其函數圖形就會一個開口向下的拋物線（下圖紅線），而其 r 就代表拋物線的曲率。世代演化一步步迭代的動作，在圖上可用一系列垂直線水平線的對應來呈現，每當有 x 值時，作一次垂直線把 x 對到函數圖線上，其高度是 y 值，接著作一次水平線對到 y=x 對角線上，此位置 y 的值會與 x 的值一樣，則再作一次垂直線觸及函數線，就相當於將代入 x 所得到的 y 再一次令成 x 再代入函數去得 y，如此循環下去，可逼近到固定點。

探討函數與對角線交會處之斜率與分叉的關係

logistic_graph.f

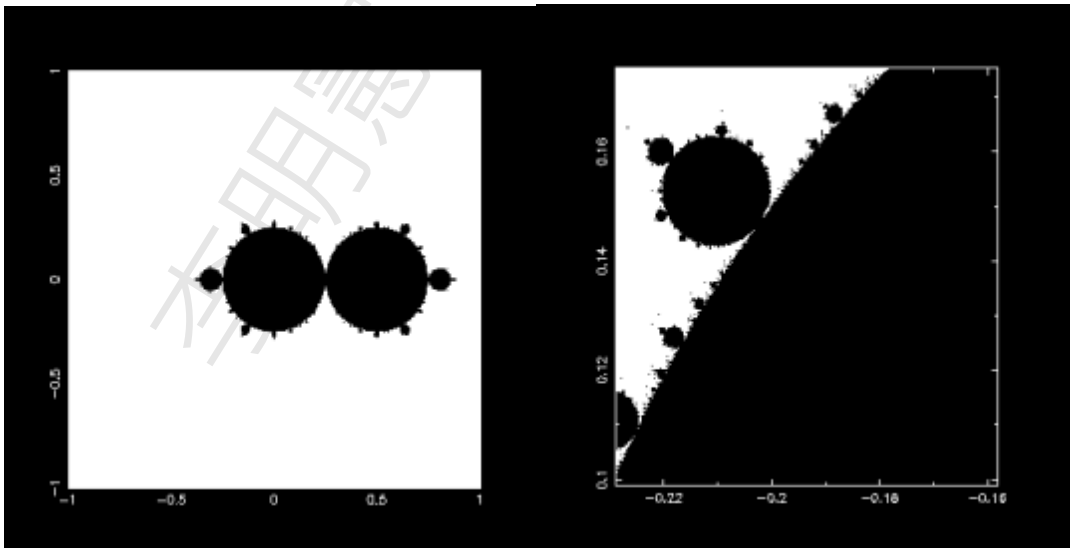
logistic_graph.x



複數 r 的 Logistic Map 與碎形

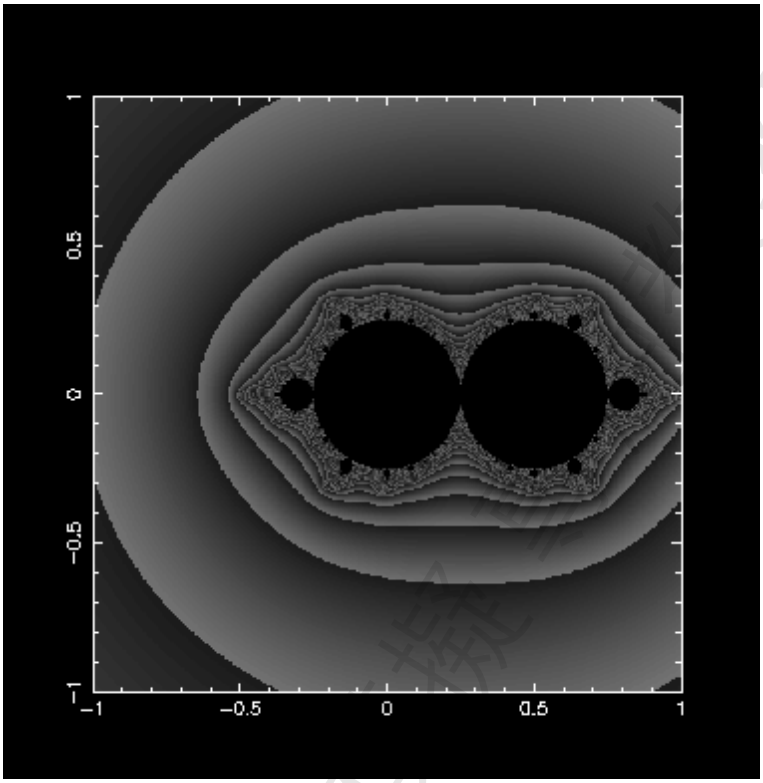
zlogistic.f

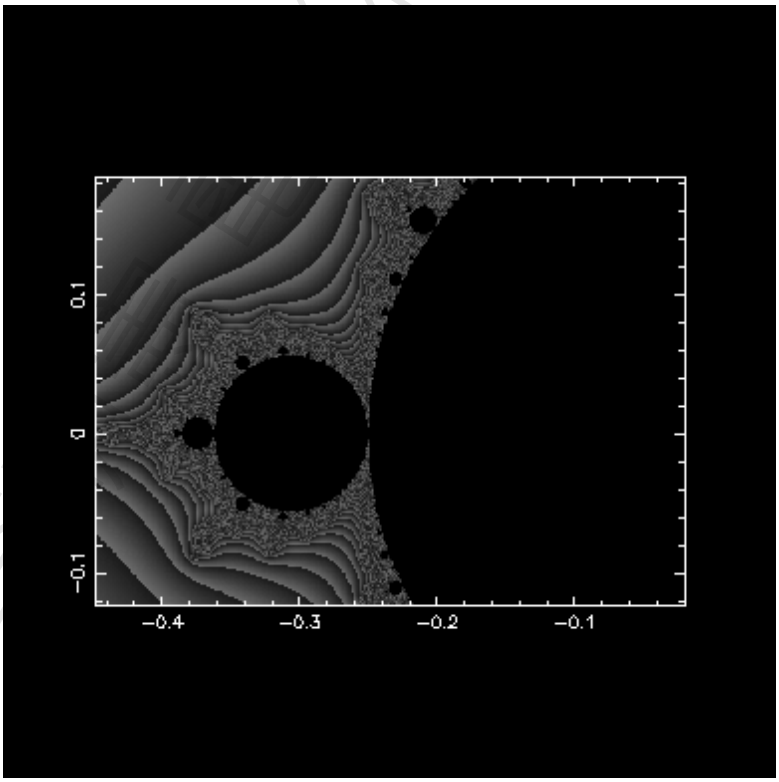
zlogistic.x

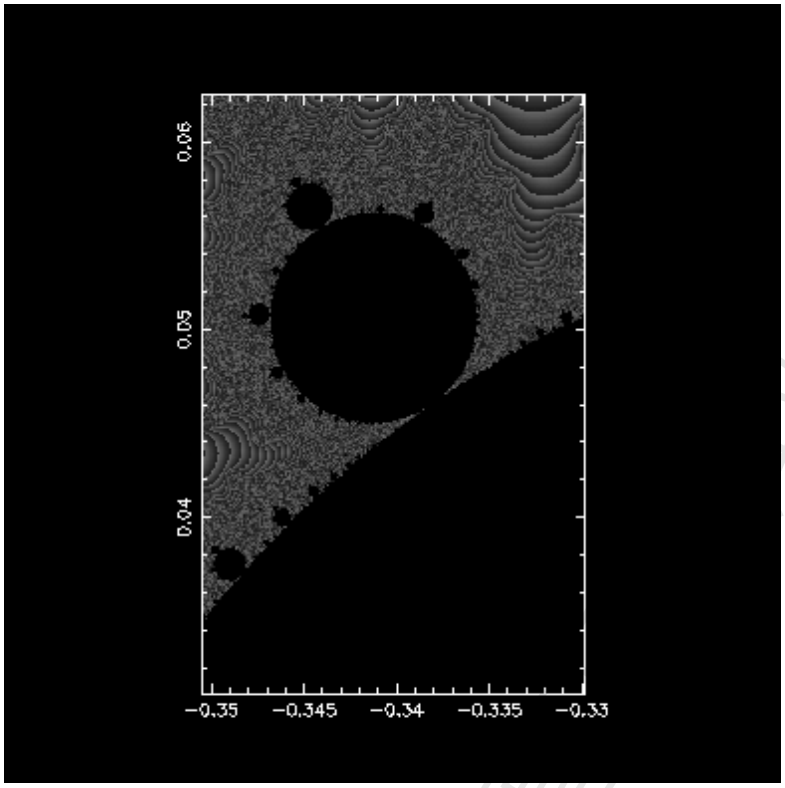


zlogistic_gray.f

zlogistic_gray.x

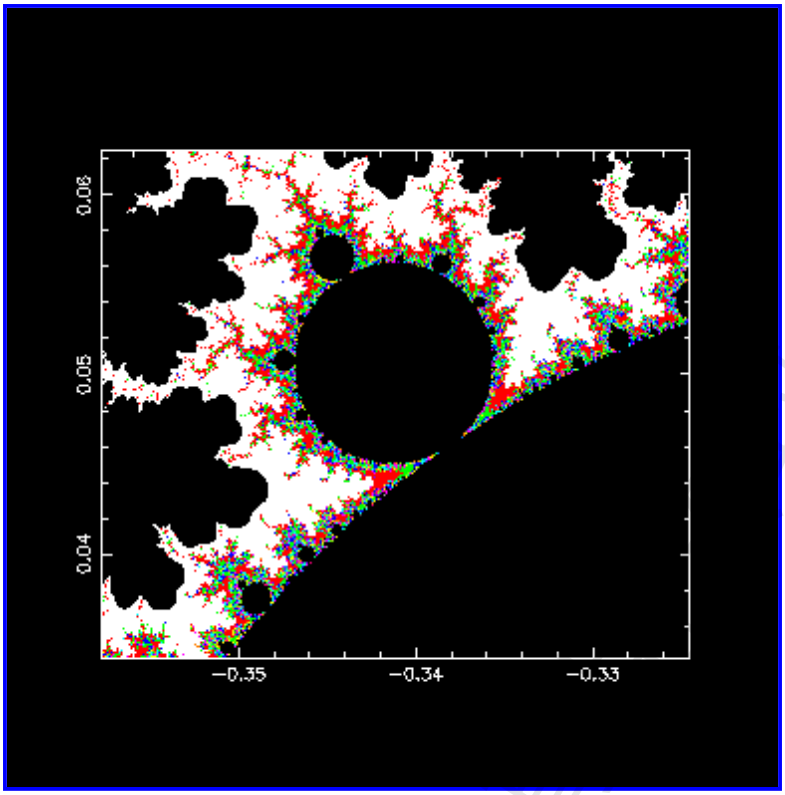






zlogistic_color.f

zlogistic_color.x



匹茲堡大學數學系的混沌課網頁（有混沌在很多不同領域的應用）
http://pear.math.pitt.edu/mathzilla/Examples/chaos/index.html

混沌藝廊
http://www-chaos.umd.edu/gallery.html

ICM 2006 Benoit Mandelbrot 碎形藝術競賽
http://www.fractalartcontests.com/2006/

碎形

（本單元範例程式全由李明憲老師撰寫，供各位同學參考）

碎形是什麼？

碎形是用來稱一種特殊的幾何圖形，它處處的局部都有著一樣的特徵結構或圖案。由於電腦的輔助。

碎形常見於不斷重覆迭代一個收縮性的函數的極限點集，叫做吸子 (attractor)，一個簡單（但不是碎形）的例子是 $f(x) = x^2$，若從 $x = [0,1)$ 出發，把 $x_{n+1} = x_n^2$ 一直代下去的話，最後的值會趨近於 0，x=0 這個點就是 $f(x) = x^2$ 這個反覆迭代映射的吸子，又如另一個例子是 $f(x) = x/2$，它最後當然是跑到 $x = 0$ 上。若是 $f(x) = -x$，則是一直在兩個點之間跳動，它們的吸子都非常簡單。

有些反覆迭代映射的吸子就複雜得多，不是一兩個固定點而已，叫奇異吸子 (strange attractor)，而它既是滿足持定函數不斷映射的結果，自然就有其一定的規律性，尤其是收縮性的映射，就是把點投射到比原來更小的靶範圍內，然而這個吸子上的所有點依定義不會再受映射對應到新的點，可以想像它就會保有局部結構下都仍一再具有此規律性。從這裏大家可以體會爲什麼經過不斷的放大去看更精細的構造時，其特徵的圖樣總會不斷出現，就是要這樣才能滿足吸子的定義。

另外，由於碎形所具有的這種特殊性質，也引發了一種定義維度的新觀點。維度可以看作是一個東西其長度的大小與重量的關係。如果其重量隨長度的平方變化，則這個件物是二維的：若是隨長度的立方變化，則是三維的。而以碎形的樣式出現的物件，其重量與長度變化的比例並不是整數，叫做是碎形維度。

大家可以透過自己動手寫程式，來畫出有趣又好玩的碎形。

碎形程式寫作要點

(1) 運算迭代

複數的使用

宣告的方法爲 `complex a` 或 `complex z(10,10)`。

複數的常數在 Fortran 裏是寫成 `(1.23, 4.56)` 這種樣子

如果需要取出其實部或虛部，可用 `real(c_num)` 或 `imag(c_num)` 這兩個內建函數； 反之，若要將一個實數填入複數中，則可用 `c_num = (1.0,0.0)*r_num` 這種方法 。

演算法

收斂度判斷

基於迴圈次數

在判定爲不收斂之後，記錄已執行的迴圈次數。

基於精確度

使用內建的對數函數 log( ) 來獲得次方

`exit` 指令

當判斷成立，並且作了適當的處理之後，我們會有需要跳出未跑完的迴圈（放棄剩下的循環不必做），這時候，可以使用 `exit` 指令來跳出一層的迴圈。它的語法極簡單，只有 `exit` 而已。

亂數

IFS 類型的碎形會需要用到亂數產生器， 普遍而言一個亂數產生器每次被呼叫就會給出一個 0 到 1 之間的不同實數，其出現的機率是相當平均的。我們這裏使用 Numerical Recipes 書中建議提供的 ran2，它均勻度的品質是最好的。另有一個 ran3，其速度比 ran2 快很多。

遞迴式的 (recursive) 副程式呼叫

有一些碎形，像是 Koch 雪花這種被叫做是 regular fractal 者，如果能用遞迴式的副程式呼叫，就能把程式寫得非常簡捷。使用者要找新版本的 Fortran 編譯器，就會支援了（實習環境中的 gfortran 是有支援的，事實上它是 fortran 95）。

(2) 圖形

paper and graph setting

紙張（圖面）大小

圖框、座標與標簽文字

point symbol type

a. coloring

cyclic index

user defined shading

b. magnification

repeating

(3) Repeating

A simple "GOTO" application

rease graph without asking

Mandelbrot 集合

最具代表性也是很多人公最認美麗的碎形，是數學家 Mandelbrot 及其他人的努力研究才引起大家的重視。它是複數平面上的點集合，叫 Mandelbrot Set，M，其定義是所有複數平面上會使反覆迭代映射 $z_{n+1} = z_n^2 + c$ 不會造成 $|z_n|$ 發散的那些 c 值。(z 的初始值要用 (0,0))

實作示範：一步一步動手寫程式（簡單範例版，不含放大及重覆）

演算法及判定（下面範例程式中的紅色部分）

設定初始 z 值（一律爲零）、設定新的 c 值（每次都不同）、反覆迭代映射（每次都檢查是否發散，若是則 c 點標成白色，並且跳出迭代迴圈）

繪圖（下面範例程式中的藍色部分）

pgopen 與 pgclos：開啓及關閉繪圖裝置

papap 與 pgenv：控制圖面的大小以及邊框範圍

pgpt：畫點

pgbbuf 與 pgebuf：開啓及關閉繪圖緩衝區

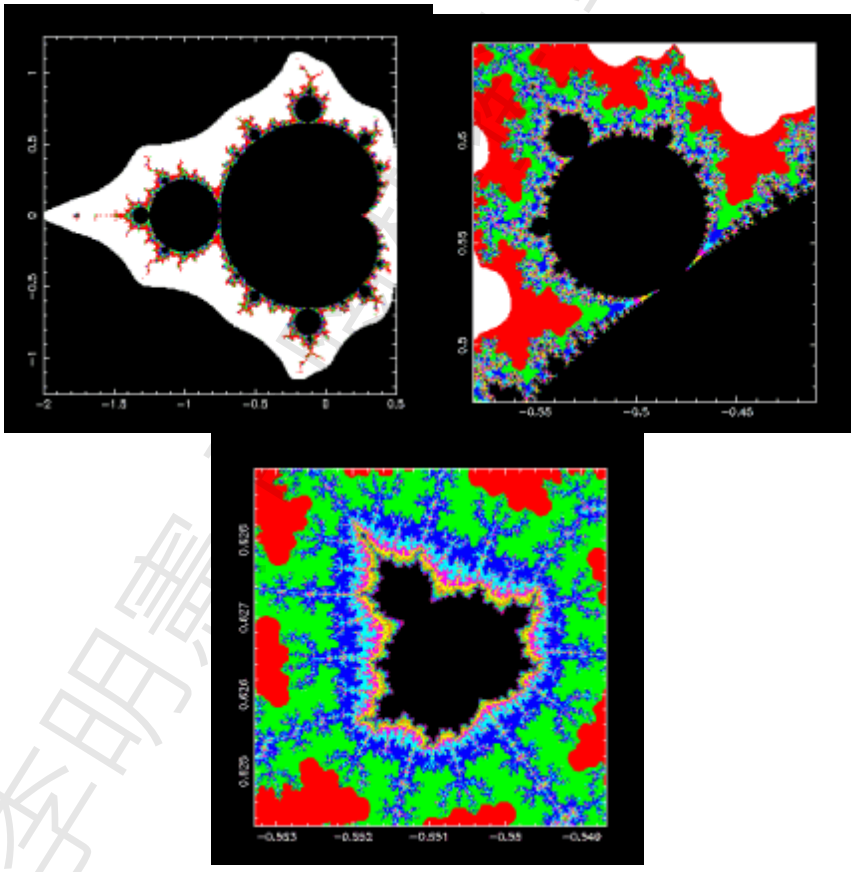
以下爲範例程式： mandel_simple.f mandel_simple.x

```
program mandelbrot_simple
implicit none
complex c,z_ini,z
integer pgopen,i,j,k,isymbol,n_gen
real c_x_min,c_x_max,c_y_min,c_y_max,c_x,c_y,abs_z

isymbol = -1

write (*,*) 'The program plot Mandelbrot set in a simple way.'
write (*,*)
z_ini = (0.0,0.0)

if ( pgopen('/xwin') .le. 0 ) stop
call pgpap(5.0,1.0)

write (*,*) 'What is the numer of iteration for the mapping ?'
read (*,*) n_gen

c_x_min = -2.0
c_x_max = 0.5
c_y_min = -1.25
```

```
c_y_max = 1.25

call pgenv(c_x_min, c_x_max, c_y_min, c_y_max, 1, 0)

do j=1,500
  c_x = c_x_min + j*(c_x_max-c_x_min)/500
  call pgbbuf
  do k=1,500
    c_y = c_y_min + k*(c_y_max-c_y_min)/500
    c = (1.0,0.0)*c_x + (0.0,1.0)*c_y
    z = z_ini
    do i=1,n_gen
      z = z*z + c
      abs_z = conjg(z)*z
      if (abs_z.gt.10e+10) then
        call pgpt(1,c_x,c_y,isymbol)
        exit
      endif
    end do
  end do
  call pgebuf
end do

call pgclos
end
```

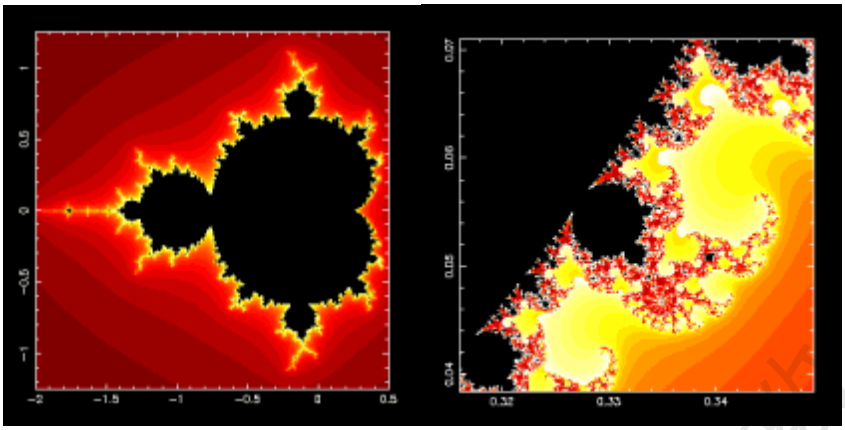
以下是完整的程式，具有放大及詢問使用者多次重畫的功能，請大家比較參考：

mandelbrot.f mandelbrot.x mandelbrot.f.txt



加入了漸層色彩指標設定的進階功能：

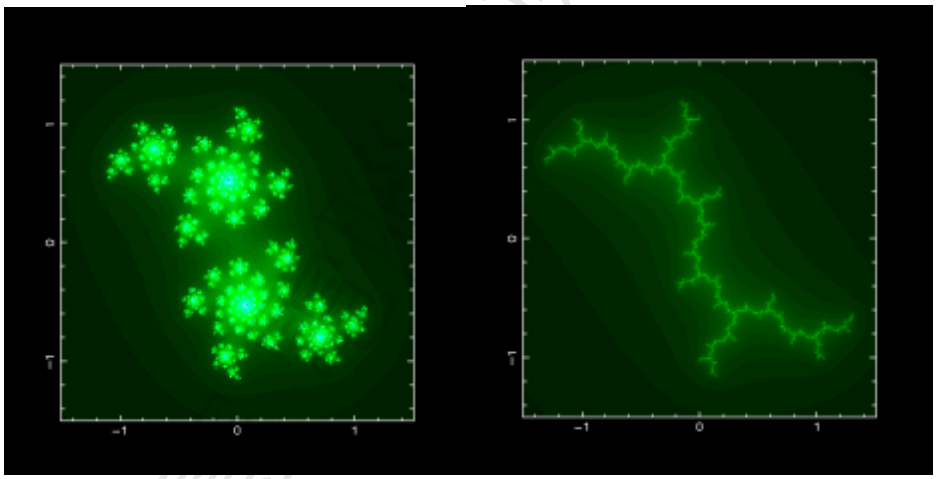
mandel_shade.f mandel_shade.x mandel_shade.f.txt



維基百科上的精采資料
http://en.wikipedia.org/wiki/Mandelbrot_set

Julia Set

Julia Set 產生的方法和 Mandelblort Set 一樣是檢驗迭代 f(z) = z*z + c 的發散性，但是是掃遍不同的 z 而不是 c。下圖左右分別是 c = (0.2,0.6) 及 c = (0.0,1.0) 的結果：



julia_shade.f julia_shade.f.txt

牛頓法

牛頓法是一種利用斜率方向的引導來求函數根的方法，這種方法在猜測點離根夠近（具體地說是在根之處的斜率與猜測點處的斜率間的變化是單調性的）時是很有效率的，然而對於不同的猜測點，從那裏出發依斜率方向去找根所會需要用到的迭代步數會比別的起始點多，在有些函數甚至會越追離根越遠。牛頓法由於具有固定的演算公式，因此若是初始猜測位置固定，處理的函數也一樣的話，

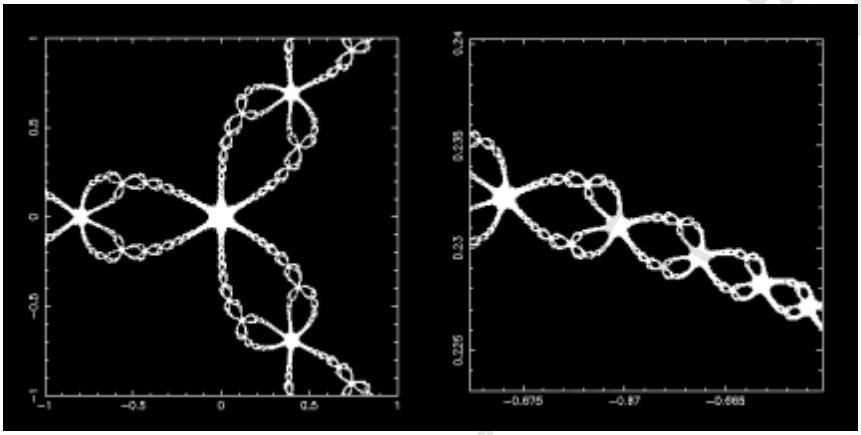
以牛頓法求複數函數 $f(z) = z^3 - 1$ 的根，就是相當於是進行

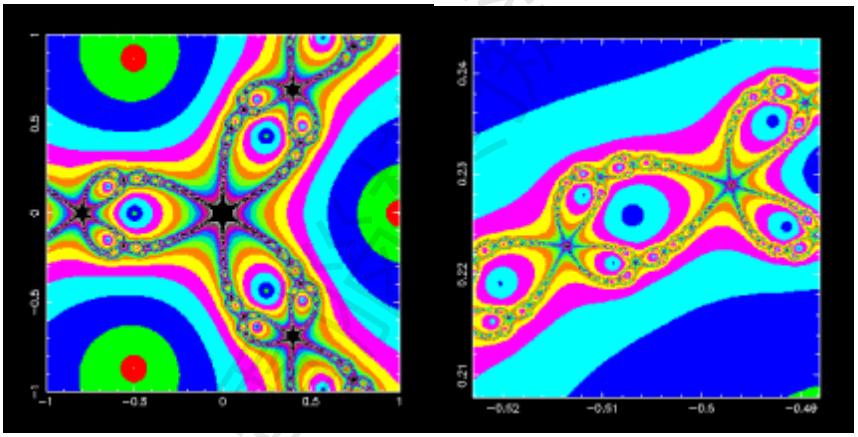
$$z_{j+1} = z_j - \frac{z_j^3 - 1}{3z_j^2}$$

的反覆迭代問題

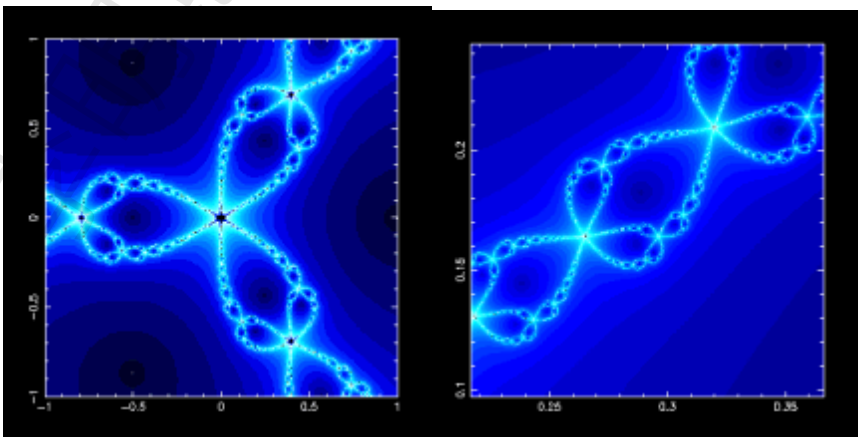
$z^3 - 1 = 0$ 要求滿足這個方程式的解，一定是具有 2p/3 對稱性的。

newton.f newton.x newton.f.txt



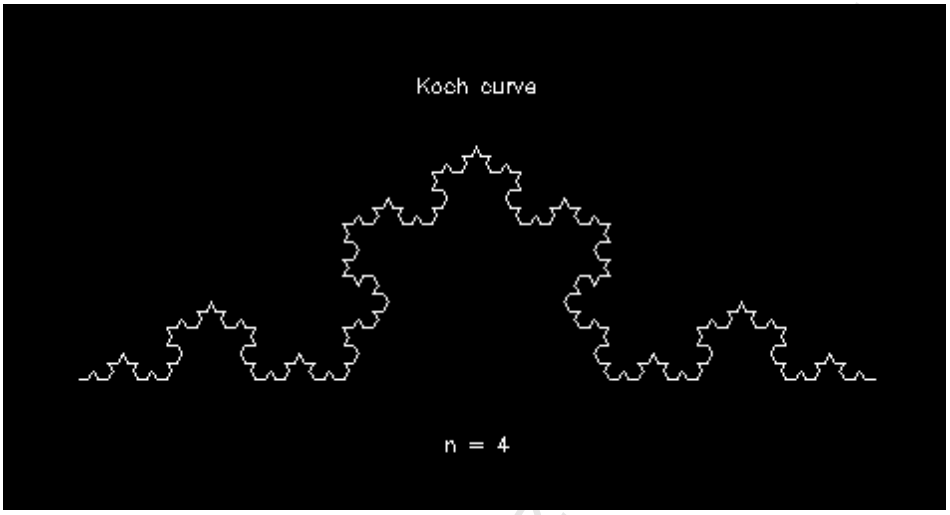


newton_shade.f newton_shade.x newton_shade.f.txt

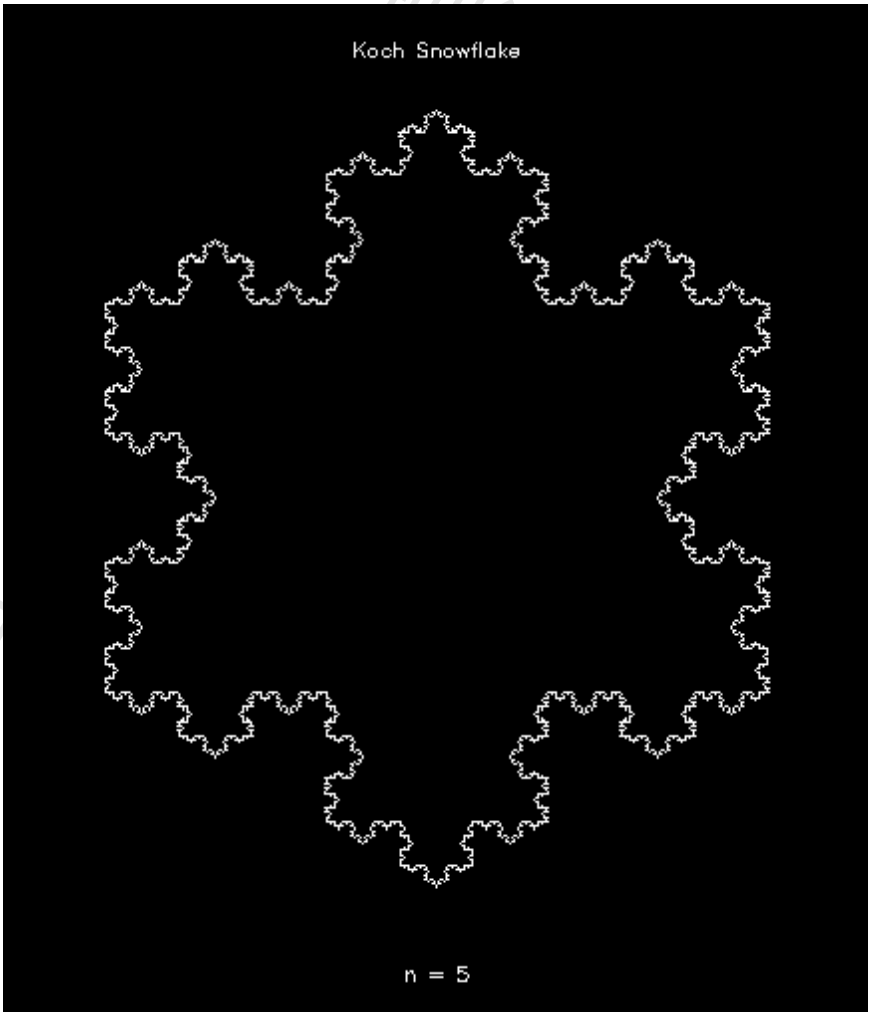


Koch 曲線

koch.f koch.f.txt koch.x



koch_snow.f koch_snow.f.txt koch_snow.x



請自己做做看以下的圖形（可利用 PGPLOT 畫多邊形與填色的功能）

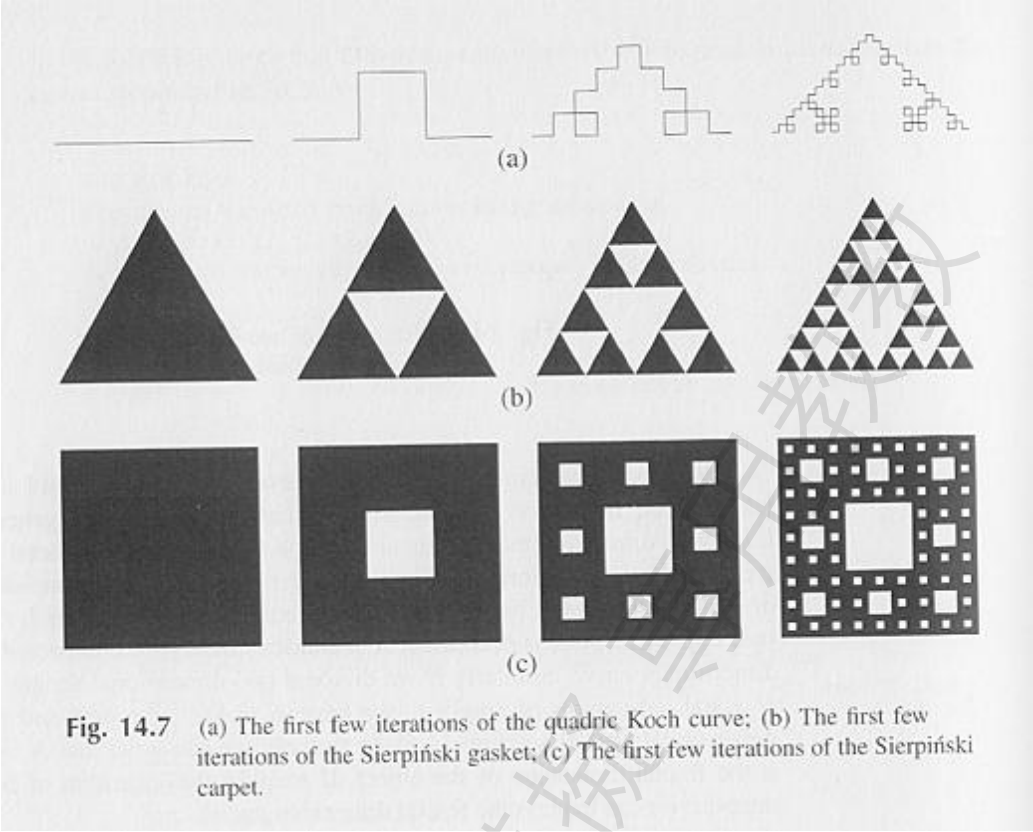


Fig. 14.7 (a) The first few iterations of the quadric Koch curve; (b) The first few iterations of the Sierpiński gasket; (c) The first few iterations of the Sierpiński carpet.

蕨葉 (Fern Leaf)

另一種產生碎形的方法是所謂的 IFS (Iterated Function Systems) ，它的吸子所構成的點集，可讓我們設計成類似蕨葉的形態，其演算法如下：

$$x_{n+1} = ax_n + by_n + e$$

$$y_{n+1} = cx_n + dy_n + f$$

a	b	c	d	e	f	p
0.00	0.00	0.00	0.16	0.00	0.00	0.01
0.85	0.04	-0.04	0.85	0.00	1.60	0.85
0.20	-0.26	0.23	0.22	0.00	1.60	0.07
-0.15	0.28	0.26	0.24	0.00	0.44	0.07

上表共有四組，最後一列的 p 是代表要使用該組演算式的機率。也可以表示成下式：

$$
\begin{aligned}
f_1(x, y) &= \begin{bmatrix} 0.85 & 0.04 \\ -0.04 & 0.85 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.00 \\ 1.60 \end{bmatrix} \\
f_2(x, y) &= \begin{bmatrix} -0.15 & 0.28 \\ 0.26 & 0.24 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.00 \\ 0.44 \end{bmatrix} \\
f_3(x, y) &= \begin{bmatrix} 0.20 & -0.26 \\ 0.23 & 0.22 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.00 \\ 1.60 \end{bmatrix} \\
f_4(x, y) &= \begin{bmatrix} 0.00 & 0.00 \\ 0.00 & 0.16 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
\end{aligned}
$$

畫出來的結果是這個樣子：

fern.f fern.x fern.f.txt

另一種，很像是路邊不知名雜草長出來的穗的部分：

weed.f weed.x weed.f.txt

另有楓葉的 IFS（請自己做做看）

a	b	c	d	e	f	p

0.14	0.01	0.00	0.51	-0.08	-1.31	0.10
0.43	0.52	-0.45	0.50	1.49	-0.75	0.35
0.45	-0.49	0.47	0.47	-1.62	-0.74	0.35
0.49	0.00	0.00	0.51	0.02	1.62	0.20

爲什麼會有自我類似？因爲這些點都是滿足迭代函數系統的吸子，也就是說這些點都滿足代了很多次也不會跑掉的特性。能夠在這樣的條件被篩選出來的點

以蕨葉點集爲例，它是由四個收縮性的映射所構成，當一個起始點被某個映射函數處理之後，就落入較小的特定範圍之中，但再緊接下來，它有可能被另一個映射函數處理，於是落入在原範圍之下的更小的另一種點分佈排列模式的範圍。你可以想像如此繼續迭代下去（過程中不同的四個函數都有用到 <問，那機率是做什麼用的？>），則每一個微小局部一定都具有類似的圖形，也就是同時具有滿足四種函數之吸子的構造。

機率可以不同

Transformation	Translation	Probability
$\begin{pmatrix} 0 & 0 \\ 0 & 0.16 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$	3%
$\begin{pmatrix} 0.85 & 0.04 \\ -0.04 & 0.85 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 1.6 \end{pmatrix}$	73%
$\begin{pmatrix} 0.2 & -0.26 \\ -0.23 & 0.22 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 1.6 \end{pmatrix}$	13%
$\begin{pmatrix} -0.15 & 0.28 \\ 0.26 & 0.24 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0.44 \end{pmatrix}$	11%

來源：http://zeuscat.com/andrew/chaos/spleenwort.fern.html

The Collage Theorem
http://www.cut-the-knot.org/ctk/ifs.shtml

Fern Functions
http://mathworld.wolfram.com/BarnsleysFern.html

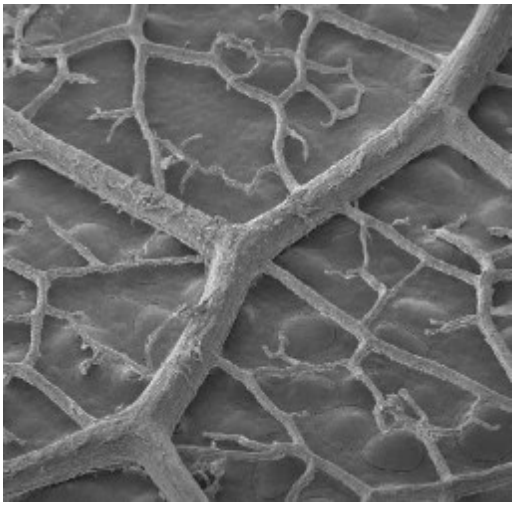
葉子的形狀
http://www.fukuoka-edu.ac.jp/~fukuhara/jikken/leaf_shape.html

葉脈網絡
http://www.nibb.ac.jp/math/work_fujita.html

碎形與自然

由於碎形所具有的自我類似的特性，使它與許多自然界中存在的幾何圖形有些相像之

處。例如枝葉的分叉、河流與葉脈的分支、山嶽的輪廓、雲的形狀、濺起的海浪、紊流與亂流，等等。可謂一沙一世界、一花一天堂。（感謝網路上之照片來源）



碎形藝術

網路上可見許多結合碎形與藝術的作品。

http://sprott.physics.wisc.edu/carlson/

http://www.fractalus.com/gumbycat/

混沌藝廊
http://www-chaos.umd.edu/gallery.html

ICM 2006 Benoit Mandelbrot 碎形藝術競賽
http://www.fractalartcontests.com/2006/

殘形美學與國畫
http://members.tripod.com/gia_5/fractal/aesth.htm

其他注意事項

本節範例程式皆使用單精度，若要達到較佳的放大級數，則應用雙精度。

參考書籍

一本很好的書是 "The Beauty of Fractals"，由 H.-O. Petigen 與 P.H. Richter 所著，Springer-Verlag 出版。

網路資源

Fractal Mathematics
http://www.hiddendimension.com/Mathematics_Main.html

Cynthia Lanius' Lessons: A Fractals Lesson - Introduction
http://math.rice.edu/~lanius/fractals/

Fractals
http://www.ocf.berkeley.edu/~wwu/fractals/fractals.html

很棒的動畫
http://www.stanford.edu/~willywu/downloads/xaos1.mpg

用 XaoS 軟體做的
http://wmi.math.u-szeged.hu/xaos/doku.php?id=main

Fractal Art
http://www.fractalus.com/info/manifesto.htm

http://math.rice.edu/~lanius/frac/

Fractals in Nature and How to Measure Them
http://www.fbmn.fh-darmstadt.de/home/sandau/biofractals/abstract_sfi.html

Julia and Mandelbrot Sets (Clark University)
http://aleph0.clarku.edu/~djoyce/julia/index.html

維基百科：Julia Set
http://en.wikipedia.org/wiki/Julia_set

維基百科：De Rham curve
http://en.wikipedia.org/wiki/De_Rham_curve

維基百科：Iterated Function System (IFS)
http://en.wikipedia.org/wiki/Iterated_function_system

利用培基 (BASIC) 語言寫碎形的網頁（外部資源）

初學者容易入門的 BASIC 語言：Decimal BASIC 官網、下載、英文網址、手冊、快速入門

BASIC 與自我類似 (Self-similarity)
http://www.geocities.jp/thinking_math_education/self-sim/self-sim.htm

BASIC 與碎形
http://www.geocities.jp/thinking_math_education/fractal/fractals.htm

True BASIC
http://www.truebasic.com/

Free BASIC Compilers and Interpreters
http://www.thefreecountry.com/compilers/basic.shtml